VŠB – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
KATEDRA INFORMATIKY

Algoritmy automatických obchodních systémů

Algorithms for Automated Trading Systems

2013 Václav Škarka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Václav Škarka**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Algoritmy automatických obchodních systémů
Algorithms for Automated Trading Systems

Zásady pro vypracování:

Automatický obchodní systém (AOS) je skupina pevně definovaných pravidel, které jednoznačně určují, kdy vstoupit na trhu, jak omezit riziko a kdy uzavírat pozice. Jedná se o počítačové programy založené na technické analýze, které na základě real-time dat z burzy určují, kdy nakoupit a prodat cenné papíry.

1. Rešerše platforem AOS.
2. Rešerše algoritmů založených na technické analýzy (indikátory RSI, MACD, MOM).
3. Implementace vybraných algoritmů, tvorba komunikačního API.
4. Simulace a modelování na historických datech, statistické vyhodnocení.
5. Závěry testování, srovnání s komerčními AOS systémy.

Seznam doporučené odborné literatury:

[1] Kendall K., Electronic and Algorithmic Trading Technology: The Complete Guide, Academic Press, 2007, ISBN 0123724910
[2] Vialar, T., Complex and Chaotic Nonlinear Dynamics: Advances in Economics and Finance, Mathematics and Statistics, Springer, 2009, ISBN 3540859772
[3] Johnson, B., Algorithmic Trading and DMA: An introduction to direct access trading strategies, 4Myeloma Press, 2010, ISBN 0956399207

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr.Ing. Michal Krumnikl**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 17. července 2013 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 17. července 2013 .....

Rád bych tímto poděkoval svému vedoucímu diplomové práce, panu Mgr. Ing. Michalu Krumniklovi za pomoc, vedení a inspiraci při její tvorbě.

Abstrakt

Cílem této práce je implementace automatického obchodního systému, který využívá obchodovací strategie založené na predikcích budoucího vývoje na burze pomocí predikčních neuronových sítí. Práce se ve své teoretické části věnuje porovnání existujících algoritmických obchodních systémů a rozebírá možnosti analýzy finančních instrumentů. Část práce je věnována vysvětlení teorie neuronových sítí a evolučních algoritmů. V praktické části je popsána, zdokumentována a otestována implementace vlastního obchodovacího algoritmu.

Klíčová slova: AOS, neuronové sítě, predikce časových řad, algoritmické obchodování

Abstract

The goal of this thesis is the implementation of automated trading system, which uses trading strategies based on predicting the future development of stock prices by neural networks. In its theoretical part, this thesis compares the existing algorithmical trading systems and explains different approaches to analysing financial data. A part of this thesis is dedicated to explaining the basic theory behind neural networks and evolutionary algorithms. The practical portion of the thesis describes, documents and tests the created trading algorithms.

Keywords: AOS, neural networks, time series prediction, algorithmic trading

Seznam použitých zkratek a symbolů

AOS	-	Automatický Obchodní Systém
API	-	Application Programming Interface
CSV	-	Comma-separated Values
EMA	-	Exponential Moving Average
HDP	-	Hrubý Domácí Produkt
HTTP	-	Hypertext Transfer Protocol
MACD	-	Moving Average Convergence Divergence
MQL	-	Metaweb Query Language
REST	-	Representational State Transfer
RSI	-	Relative Strenght Index
URI	-	Uniform Resource Identifier
VB.NET	-	Visual Basic .NET

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Automatické obchodní systémy jsou platformami, které umožňují automatizovat obchodování na burze s cennými papíry a dalšími finančními instrumenty dle určité strategie. Tyto systémy si v posledních letech získávají rostoucí podíl na všech prováděných obchodech a algoritmické obchodování je již mnohdy dominantním způsobem provádění obchodů na akciovém trhu [1].

Těchto systémů existuje velké množství a pro tvorbu svých strategií využívají široké spektrum metod a postupů, které proto stojí za prozkoumání. V rámci této práce se podíváme na vzorek nejpopulárnějších existujících platforem pro automatické obchodování a představíme si některá jejich specifika. Dále rozebereme rozdílné typy analýzy finančních instrumentů, které lze provádět pro vyhodnocení toho, kdy je vhodné vstoupit do určité pozice na trhu a kdy z ní vystoupit pro maximalizaci zisku. Nejpopulárnější z těchto typů analýz je v kontextu AOS technická analýza, která slouží jako základ mnoha automatických obchodních strategií [2].

Ačkoliv ji lze považovat za dostatečný nástroj pro analýzu trhu a vytváření obchodních rozhodnutí, obsahuje zároveň omezení (zejména jednoduchost poskytovaných signálů), kvůli kterým má smysl zkoumat i jiné způsoby, kterými by bylo možné předvídat vývoj situace na trhu.

Jedním z těchto způsobů je využití neuronových sítí. Protože vývoj cen finančních instrumentů na trhu lze vyjádřit jako časovou řadu hodnot cen dodržující určité trendy a zákonitosti, je možné adaptovat neuronové sítě pro popis průběhu této časové řady a následně využít pro tvorbu předpovědí jejího budoucího vývoje.

Součástí této práce je implementace obchodovacích algoritmů založených na neuronových predikčních sítích a genetických algoritmech. Pomocí těchto algoritmů dokáže implementovaný systém generovat obchodní příkazy odpovídající předpokládané budoucí situaci na trhu.

Při použití neuronových sítí pro predikci časové řady akciových cen, které se místy nevyhnutelně vyvíjí nepředpokládatelným způsobem, je nutné překonat několik implementačních výzev, které jsou v rámci této práce rozebrány a analyzovány. Finální algoritmus pak v závěru práce bude otestován v rámci simulace obchodování nad historickými daty.

Tato práce v rámci následující kapitoly představí koncepty a vybrané zástupce automatických obchodních systémů. Následně budou vysvětleny přístupy k analýze finančních trhů a detailněji rozebráno několik metod v rámci technické analýzy. V šesté a sedmé kapitole jsou představeny teoretické základy predikčních neuronových sítí a evolučních algoritmů použitých pro jejich učení. Osmá kapitola je věnována obsáhlé implementační části práce a její testování na reálných datech je rozepsáno v kapitole deváté. Celá práce končí vyhodnocením provedených testů a zhodnocením implementovaného obchodovacího algoritmu.

2 Automatické obchodní systémy

Automatické obchodní systémy jsou počítačové programy, které vyhodnocují aktuální situaci na trhu s určitým finančním instrumentem a dle přednastavených pravidel automaticky rozhodují o nákupu/prodeji daného instrumentu. V roce 2006 byla celá třetina všech akciových obchodů na burze v Evropské Unii a Spojených Státech Amerických provedena právě automatickými obchodními systémy [3]. Do roku 2009 toto procento vzrostlo a například na americkém trhu dosáhlo téměř tří čtvrtin všech provedených obchodů [4].

Obchodní algoritmy těchto systémů jsou většinou vyvíjeny a testovány nad sadou historických burzovních dat, u kterých je známý následný vývoj trendu cen. Součástí automatických obchodních jsou rozhraní pro zobrazování grafických vizualizací dat obchodovaných instrumentů a nástroje na jejich technickou analýzu.

2.1 Vybrané automatické obchodní systémy

Automatických obchodních systémů, mezi kterými mohou uživatelé volit, je k dispozici obrovské množství. Jejich licenční politika je ovšem mnohdy velmi prohibitivní a není v silách této práce udělat rešerši všech velkých dostupných systémů. Proto je analýza existujících aplikací zaměřena na malou hrstku platforem, které si kolem sebe vybudovaly početnou vývojářskou a uživatelskou komunitu a díky tomu poměrně viditelnou pozici v prostředí internetu [5].

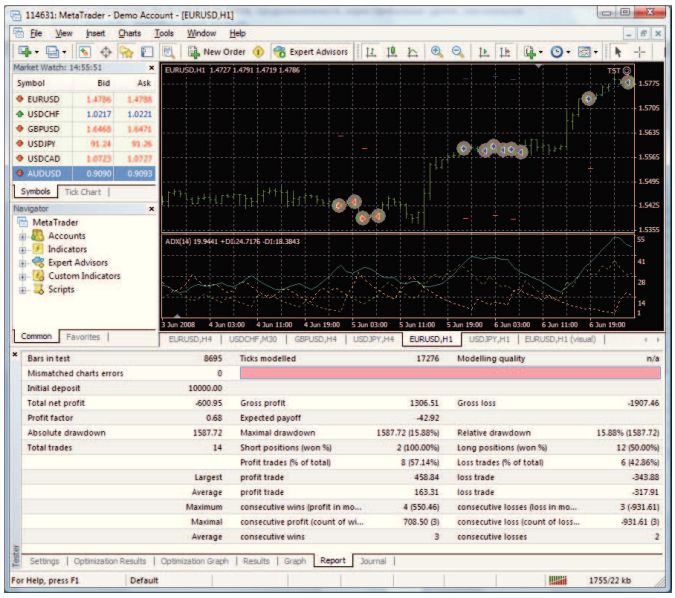
2.1.1 MetaTrader 4

MetaTrader[1] 4 je automatická obchodní platforma vyvinuta v roce 2005 společností MetaQuotes Software. Od té doby se software stal velmi populárním zejména na Forex trzích, kde se obchodují měnové páry. MetaTrader je licencován devizovým makléřům, kteří ho pak poskytují svým klientům. Aplikace je z toho důvodu rozdělena na serverovou část (operovanou makléřem), a klientskou část, která umožňuje reálné zobrazování obchodních informací, analýz a vytváření obchodních pokynů. Obě části oficiálně podporují pouze operační systém Windows, ačkoliv existují neoficiální makléřské nadstavby pro Mac OS.

Silnou stránkou platformy je možnost psát a využívat vlastní obchodovací skripty (v jazyce **MQL4)** pro automatizaci obchodování. Obdobně lze pomocí skriptovacího jazyka definovat zcela nové indikátory. Kolem této možnosti rozšiřovat MetaTrader se vytvořila rozsáhlá komunita vyvíjející a distribuující open source knihovny a obchodovací sub-aplikace.

Od roku 2010 je k dispozici nová verze MetaTrader 5, ale i díky velmi aktivní open source komunitě je přechod uživatelů od verze 4 pomalý a původní aplikace je v době psaní této práce stále populárnější, než její pokračování.

[1] Dostupný na adrese http://www.metatrader4.com/



Obrázek 1: MetaTrader 4

2.1.2 MultiCharts

MultiCharts[2] je systém, jehož první verze vyšla již v roce 2004 a od té doby pro něj výrobce MultiCharts LLC. vydal již 7 navazujících verzí. Podobně jako předchozí zmíněný systém, i MultiCharts je oficiálně určený pouze pro Windows a kromě zadávání automatických obchodních příkazů umožňuje i provádění technické analýzy nad obchodními daty. Také umožňuje rozšíření základních obchodních strategií a standardních indikátorů o vlastní, uživatelsky naprogramované položky. Ty jsou programovány vlastním programovacím jazykem **PowerLanguage**, který mj. umožňuje využití externích dynamických knihoven. Díky tomu v rámci něj lze použít některý z populárních vysokoúrovňových programovacích jazyků, jakými jsou například C# a C++. PowerLanguage je kompatibilní se skripty napsanými v jazyce **EasyLanguage**, který slouží jako skriptovací prostředek k dalšímu z populárních obchodních systémů – **Tradestation**.

Z programátorského hlediska je zajímavá existence odlehčené verze MultiCharts .NET, která nabízí limitovanou funkcionalitu původní aplikace zcela zdarma a je upravena tak, že jako skriptovací prostředky využívá množinu .NET (C#, VB.NET) jazyků.

[2] Více na adrese http://www.multicharts.com/

Obrázek 2: Odlehčená verze MultiCharts .NET

2.1.3 TradeStation

TradeStation[3] je jedna z nejstarších platforem pro automatické obchodování a společnost Omega Research, Inc. (dnes známá už jen jako TradeStation) ji na trh poprvé uvedla v roce 1991. První verze, která uživatelům nabídnula možnosti automatizace obchodů, byla TradeStation 6 vydaná v roce 2001. Kromě standardních možností analýzy a obchodování, nabízí aplikace vlastní programovací jazyk **EasyLanguage** pro tvorbu uživatelských skriptů, indikátorů a obchodních strategií. Skripty vytvořené v EasyLanguage jsou přenosné a použitelné v některých aplikacích ostatních výrobců (např. v již zmíněném MultiCharts).

Unikátní součástí platformy je rozhraní **WebAPI**, které je navrženo jako webová služba ve stylu architektury REST a pohání iOS a webové aplikace. Zároveň je možné k tomuto rozhraní přistupovat cizími uživateli v rámci aplikací třetích stran, které mohou s rozhraním komunikovat přes HTTP požadavky.

2.1.4 Wealth Lab

Wealth Lab[4] je systém, který je na trhu od roku 2000 a svým uživatelským prostředím je cílený více na uživatele se znalostí programování. Je postavený nad .NET frameworkem a obsahuje integrované vývojové prostředí založené na syntaxi jazyka C#, které uživatelům umožňuje vytvářet obchodní strategie založené na hodnotách technické analýzy finančních instrumentů. Kromě psaní zdrojových kódů těchto skriptů umožňuje Wealth Lab programování na základě přetahování jednoduchých vstupních a výstupních pravidel do sekvence, což zpřístupňuje tvorbu vlastních, jednodušších strategií i uživatelům bez hlubší znalosti programovacích jazyků.

[3] K dispozici na adrese http://www.tradestation.com/
[4] K dispozici na adrese http://www.wealth-lab.com/

3 Analýza burzovních instrumentů

Ke zkoumání závislostí a vývoje finančních instrumentů na trhu lze využít velké množství metod a postupů. Obecně se tyto metody dělí na tři kategorie, podle toho, který aspekt trhu analyzují pro určení dalšího vývoje cen. Těmito kategoriemi jsou fundamentální analýza, technická analýza a psychologická analýzy.

3.1 Fundamentální analýza

Fundamentální analýza akcie je přístup založený na pozorování a ohodnocování základních kurzotvorných faktorů a vlastností, které ovlivňují současný byznys sledované společnosti (případně odvětví, nebo globální ekonomické situace) a její potenciál do budoucna. Na rozdíl od technické analýzy nebere fundamentální analýza v potaz krátkodobé výkyvy cen akcie a místo toho se zaměřuje na vyhodnocení dlouhodobého „zdraví“ sledované společnosti.

Informace získané a využité fundamentální analýzou lze rozdělit na **kvalitativní** a **kvantitativní**. Kvalitativní informace pokrývají faktory, které se nedají vyjádřit číselně a jež je obtížné korektně analyzovat. Základními otázkami, jejichž zodpovězení vede ke kvalitativním fundamentálním informacím, jsou: „**Jaký je business model společnosti? Na čem společnost vydělává a jak?**“, „**Jaké konkurenční výhody má společnost na své straně?**“, nebo „**Jak jsou nastavené vztahy mezi vedením společnosti a jejími akcionáři?**“.

Kvantitativní informace jsou získávány z veřejně přístupných finančních výkazů a statistických dat. Tento typ informací zahrnuje hodnoty jako je velikost dividendy vyplácená investorům, vývoj ekonomiky vyjádřený HDP, úroková míra, inflace, účetní hodnota akcie, zisk společnosti a další. Z těchto hodnot je v rámci fundamentální analýzy vypočítána řada indikátorů, které popisují, jak výnosné/perspektivní/bezpečné je investování do daného instrumentu [6].

3.2 Psychologická analýza

Psychologická analýza je od ostatních přístupů k investování na finančních trzích velmi odlišná v tom, že se nesoustředí na zkoumání hodnoty a zdraví finančních instrumentů, ale na to, jak tyto instrumenty vidí investoři a jak se při jejich obchodování chovají. Účastnící obchodování na finančních trzích jsou rozdělování do skupin podle typu do svého chování. Těchto rozdělení je celá řada dle rozdílných metodologií a zahrnují role, jakými jsou spekulanti, hráči, tradeři a další. Psychologická analýza vychází z toho, že v davu se ztrácí vlastnosti individuálních jedinců a vytváří se nové, davem určené. Díky tomu lze zkoumat a předpovídat investiční chování jednotlivých skupin účastníku trhu a jejich vliv na vývoj ceny instrumentu [7].

3.3 Technická analýza

Technická analýza je přístupem, který předpovídá směr vývoje cen finančních instrumentů na základě analýzy historických burzovních dat (zejména pak uzavíracích cen, cenových minim a maxim, objemů). Tento typ analýzy využívá řadu rozdílných nástrojů a postupů, základními z nichž jsou **analýza vzorců v grafu cen** a **analýza technických indikátorů** [8]. Všechny postupy obecně vedou buď k potvrzení existujícího cenového trendu na trhu, k signalizaci beztrendového období, nebo k identifikaci práce se měnícího trendu.

V rámci analýzy vzorců v grafu se vychází z faktu, že chování cen na burze vytváří v grafech vývoje těchto cen určité obrazce a vzory, které se dlouhodobě opakují. Analýza technických indikátorů pak vyhodnocuje trh z pohledu ukazatelů, které jsou vypočítány různorodými matematickými transformacemi historických cen a objemů. Technická analýza je většinou porovnávána s výsledky analýzy fundamentální a až na základě vzniklého kontrastu jsou vytvářeny obchodní rozhodnutí.

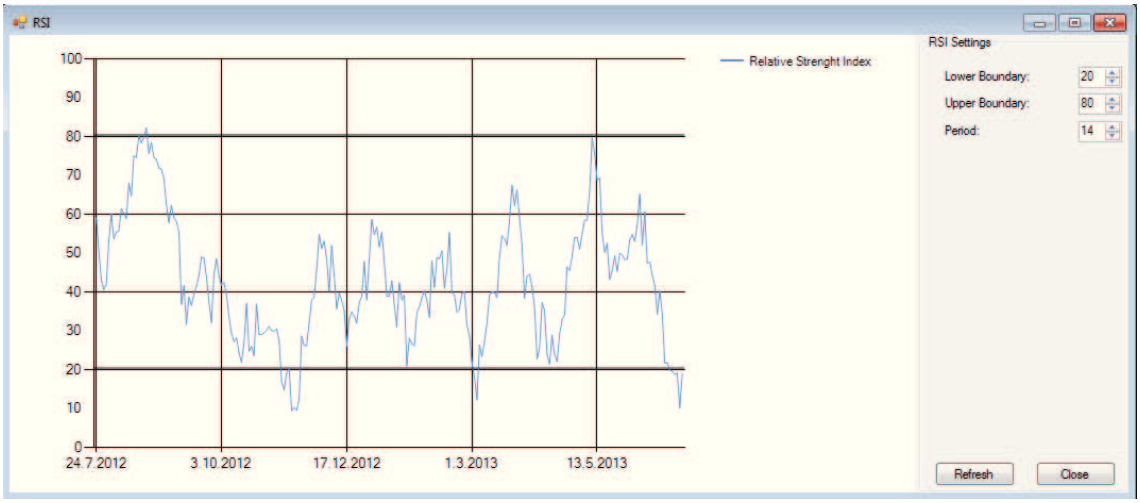
4 Vybrané indikátory technické analýzy

Technických indikátorů je k dispozici velké množství a stále vznikají nové. Obecně je lze rozdělit na dvě skupiny. Tou první jsou indikátory, které předpovídají změnu **dlouhodobého** trendu, před tím, než se jí přizpůsobí chování investorů. Příkladem indikátoru náležící do této skupiny může být třeba indikátor *MACD* (klouzavý průměr). Druhou skupinou jsou indikátory, které mapují právě probíhající **krátkodobý** trend a potvrzují jeho existenci a případně sílu tohoto trendu. Jedním z indikátorů patřících do této skupiny je například *Momentum* [8].

V rámci implementační části této práce byla do programu začleněna malá množina vybraných indikátorů, které jsou považovány na základní a nejtradičnější technické ukazatele a jejichž kombinace dává uživateli dostatečné možnosti analyzovat stávající situaci na trhu se sledovaným finančním instrumentem. Vybraná množina obsahuje indikátory obou skupin, tedy ty potvrzující trend i ty předpovídající budoucí změny.

4.1 Index relativní síly (RSI – Relative Strenght Index)

Index relativní síly (dále jen RSI) je oscilátorem pohybujícím se mezi limitními hodnotami 0 a 100. Ukazatel RSI poměřuje velikost a rychlost tempa změn ceny pozorovaného aktiva. Rychle rostoucí cena aktiva se projeví vysokou hodnotou RSI, zatímco klesající cena způsobí nižší hodnoty indexu. RSI využívá toho, že cena pozorovaného aktiva se většinu času (80%) pohybuje mezi hodnotami 20 a 80, tedy mimo pásmo extrémních výkyvů cen a trendů. Pokud jsou tyto hodnoty překročeny, vysílá RSI jasný signál toho, že se s cenou pozorovaného aktiva děje něco netypického [9].



Obrázek 3: Indikátor RSI

Při hodnotě RSI nad 80 je signalizována překoupenost aktiva, při hodnotě pod 20 naopak přeprodanost. Hraniční hodnoty 20/80 lze zaměnit za širší hranice (10/90), nebo užší hranice (30/70). Užší hranice činí RSI citlivějším na změny. Indikátor vysílá signál k prodeji/nákupu aktiva častěji, ale tento signál je slabší a méně spolehlivý. Naopak u širších hranic RSI dojde k vyslání signálu jen v situacích s výjimečně silným tempem změny ceny aktiva. Tento signál relativně spolehlivě popisuje situaci, ale není tak citlivý a proto přichází pozdě. Vzorec pro výpočet je daný formulí uvedenou níže.

$$RSI = 100 - \frac{100}{1 + RS}$$

$$RS = \frac{EMA(Up, n)}{EMA(Down, n)}$$

Základním předpokladem pro výpočet RSI je fakt, že uzavírací cena aktiva v každý den je vždy přírůstkem (proměnná *Up*), nebo úbytkem (proměnná *Down*) oproti otevírací ceně daného dne. Z těchto přírůstků a úbytků se vytvoří exponenciální klouzavý průměr za *n* dnů. Typicky se pro výpočet RSI používá rozsah 14-ti dnů. Tento rozsah *n* lze měnit pro tvorbu kratších, respektive delších předpovědí, případně pro analýzu týdnů a měsíců namísto dnů.

Nedostatkem výše zmíněného vzorce je využití exponenciálního klouzavého průměru, jehož výsledek je závislý na tom, jak velké úseky historických dat jsou s ním poměřovány. O něco konzistentnější výsledky pak vrací upravený vzorec RSI (Cutlerův RSI), který využívá pouze jednoduchého klouzavého průměru.

$$RS = \frac{SMA(Up, n)}{SMA(Down, n)}$$

Obě varianty vrací podobné hodnoty a liší se zejména intenzitou vysílaných signálů.

4.2 MACD – Moving Average Convergence/Divergence

MACD je indikátorem, který signalizuje změny v síle, směru, tempu změn a trvání trendu. Toho je docíleno sledováním konvergence (sbíhavosti) a divergence (rozbíhavosti) klouzavých průměrů [10]. Oscilátor MACD využívá tří hodnot, resp. křivek:

- 1. hodnota je takzvanou MACD křivkou a je určena rozdílem exponenciálních klouzavých průměrů. Jeden z těchto průměrů je tzv. „*rychlý*“ – spočítaný z pohybu cen v krátkém časovém období. Druhý je „*pomalý*“ vypočítaný z pohybu cen v dlouhém období.

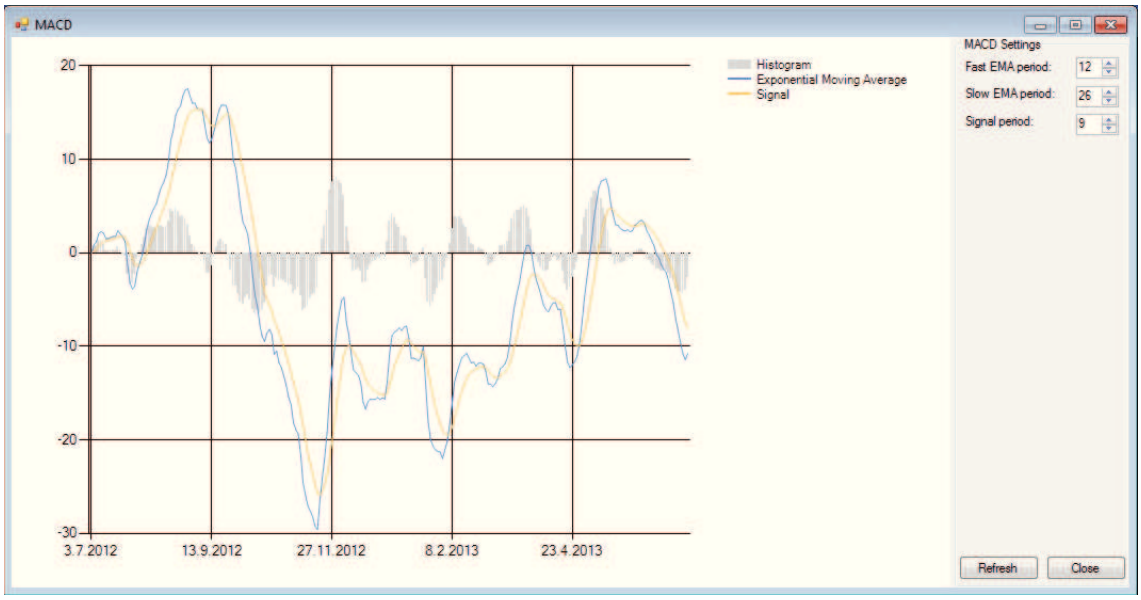
$$MACD = EMA(12) - EMA(26)$$

- 2. hodnota je tzv. „*Signální křivka*“, která je definována jako průměr MACD křivky za krátké období (typicky 9 dnů)

$$Signal = EMA(MACD, 9)$$

- 3. Hodnota je rozdíl MACD a signální křivky zobrazený v podobě histogramu.

$$Histogram = MACD - Signal$$



Obrázek 4: Indikátor MACD

4.2.1 Obchodování podle MACD

Indikátor MACD vysílá tři typy signálů v následujících situacích:

- Překřížení MACD a signální křivky
  - Toto překřížení signalizuje změnu aktuálního trendu (aproximovaného signální křivkou). Pokud MACD protne signální křivku zdola, indikátor vysílá signál k nákupu a předpovídá budoucí růst cen aktiva.
  - Obdobně, pokud MACD překříží signální křivku shora, indikátor signalizuje budoucí pokles ceny aktiva.
  - Body překřížení lze vypozorovat z histogramu, který znázorňuje rozdíl mezi MACD a signální křivkou. V místech (dnech), kde je hodnota histogramu rovna nule došlo k překřížení křivek.
- Překřížení MACD křivky a nulové linie
  - Tento signál funguje obdobně jako přechozí příklad. Neupozorňuje však většinou na změnu trendu, ale slouží jako důkaz o existenci a trvání stávajícího trendu.
  - Matematicky je tato situace definována jako:

$$EMA(12) - EMA(26) = 0$$

  - Pokud k této situaci dojde překřížením MACD křivky a nulové linie shora, je potvrzen dlouhodobý medvědí trend. A naopak překřížením zdola je potvrzen býčí trend.
- Konvergence/divergence mezi cenou aktiva a signálem MACD
  - Rozeznáváme medvědí a býčí divergenci.

- Medvědí divergence znamená situaci, kdy cena sledovaného aktiva vytvořila nové maximum, které převyšuje maximum z přechozího sledovaného období (dne/týdne/měsíce). Zároveň došlo k tomu, že indikátor MACD vytvořil nižší maximum, než je předchozí hodnota.
- V případě této situace je vyslán signál o změně trendu na trhu z býčího na medvědí (ceny aktiva budou dále klesat).
- Obdobně je tomu u býčí divergence, která signalizuje opačný stav a dochází k ní, pokud cena aktiva vytváří větší minima, ale MACD indikátor se otáčí a vytvoří menší minimum, než v předchozí periodě.

4.3 Momentum

Momentum je indikátorem, který má za cíl odstranit nedostatky a slabá místa vyrovnávacích průměrů a zejména pak opožděnost jimi vysílaných signálů [8].
Momentum je definováno vzorcem:

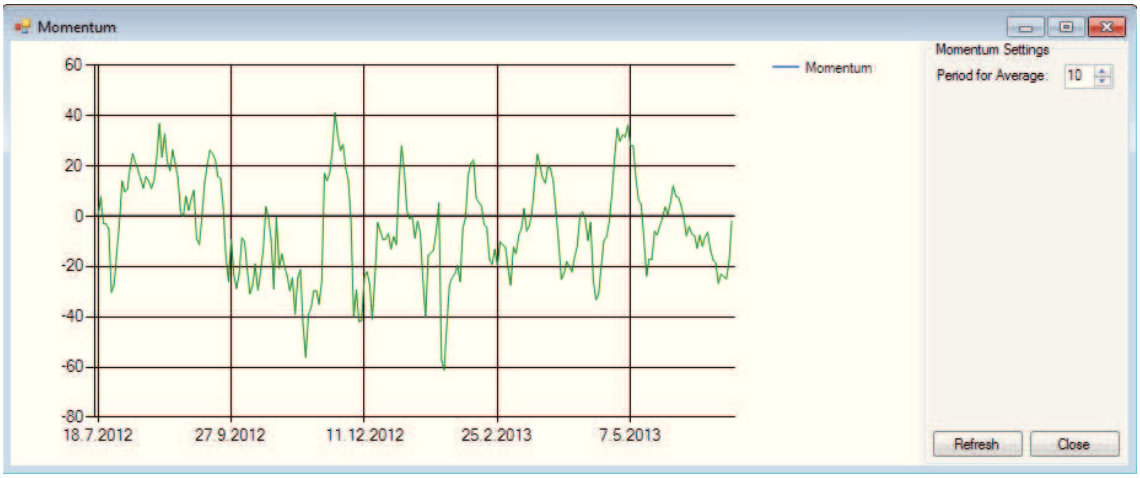
$$Mom = ZC - ZC(x)$$

Momentum je tedy rozdílem uzavírací ceny sledovaného dne od průměrné uzavírací ceny za poslední období x dnů. Čím je tento rozdíl větší, tím silnější signál momentum vysílá.
Spolehlivost momenta výrazně ovlivňuje zvolená perioda x. Při zvolení příliš krátké periody vysílá momentum nespolehlivé signály. Při zvolení příliš velkého x pak pro změnu indikátor nedokáže adekvátně zareagovat na všechny větší výkyvy v rámci sledovaného období. Typicky se proto volí období deseti pracovních dnů jako adekvátně spolehlivé [8].

Momentum vysílá dva typy signálů:

- Pokud je hodnota momenta menší, než nula, je aktuální cena menší, než je průměr předchozích x dnů. Indikátor vysílá signál, že cena aktuálně klesá.
- Pokud je současná cena vyšší, než je průměr sledovaného období, vysílá indikátor signál, že cena aktiva roste.

Velikost momenta dále určuje, jak silný je současný trend, který je tímto indikátorem potvrzován.



Obrázek 5: Indikátor Momentum

4.4 Stochastik

Stochastik je indikátorem na bázi oscilátoru, který se pohybuje mezi hodnotami 0 a 100 a signalizuje, zda je pozorované aktivum ve stavu překoupenosti, nebo přeprodanosti [11].

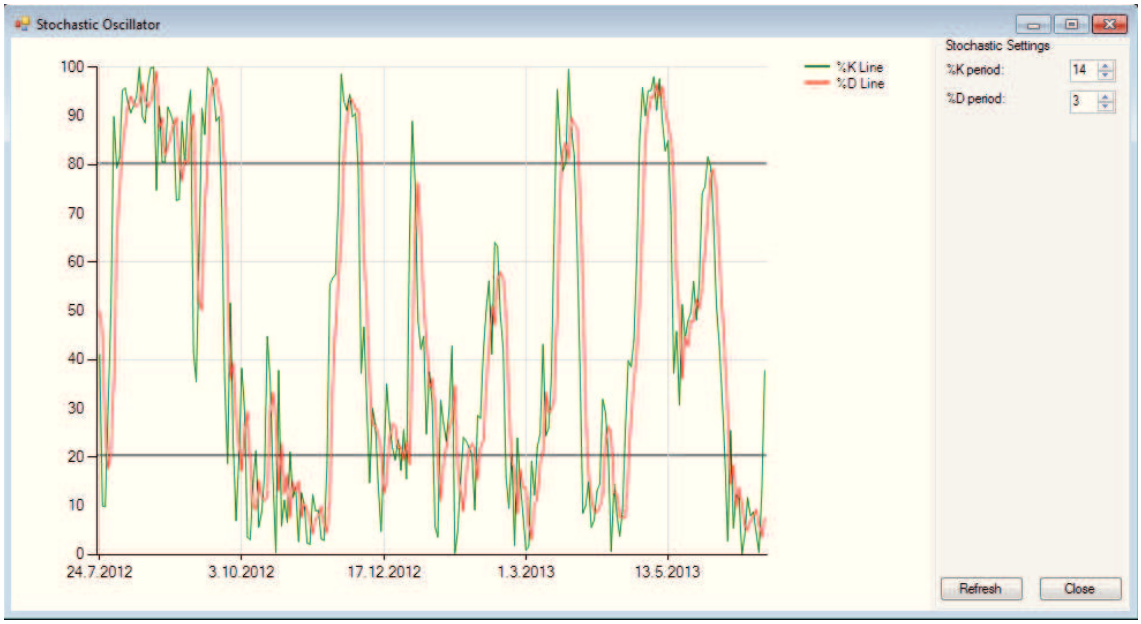
Tento indikátor využívá toho, že je-li uzavírací cena aktiva poblíž maxima uplynulé periody, aktivum má tendenci i nadále růst. Naopak pokud se tato cena pohybuje okolo minima sledované periody, má aktivum tendenci klesat. Stejně jako u jiných indikátorů lze i stochastickým oscilátorem sledovat delší časové periody, než jsou jednotlivé dny (týdny, měsíce). Indikátor se skládá ze dvou křivek: *%K* a *%D*, které nabývají hodnot 0 až 100 (procent) v čase *x*.

Křivka *%K* je definována následujícím vzorcem, kde proměnná *C* je aktuální uzavírací cenou, *Low* je nejnižší cenou za posledních *n* period (typicky 14 dnů) a *High* vyjadřuje nejvyšší cenu za shodný počet *n* posledních period.

$$\%K = (C - Low)/(High - Low) * 100$$

Křivka %D je vypočítána jako exponenciální klouzavý průměr křivky *%K* za tři periody.

$$\%D = EMA_3(\%K)$$



Obrázek 6: Stochastický oscilátor

4.4.1 Obchodování podle Stochastického oscilátoru

Stochastik vysílá signály, pokud dojde k divergenci/konvergenci vývoje křivek %D a %K. Protože obě křivky se vzájemně protínají poměrně často, je sledovaná oblast omezena na horní a dolní extrémy. Tedy na oblast, ve které hodnoty %D a %K vyjadřuje pohyb ceny poblíž minima, nebo maxima sledovaného časového období a tedy naznačuje přeprodanost, nebo překoupenost aktiva.

Konkrétně vysílá Stochastik následující signály:

- Pokud se obě křivky nacházejí v oblasti překoupenosti (horní spektrum hodnot) a křivka %K překříží křivku %D shora, vysílá stochastik signál o budoucím poklesu ceny aktiva.
- Pokud se obě křivky naopak nacházejí v oblasti přeprodanosti (dolní spektrum hodnot) a křivka %K překříží křivku %D zdola, vysílá indikátor signál o budoucím růstu ceny aktiva.

4.5 Aroon

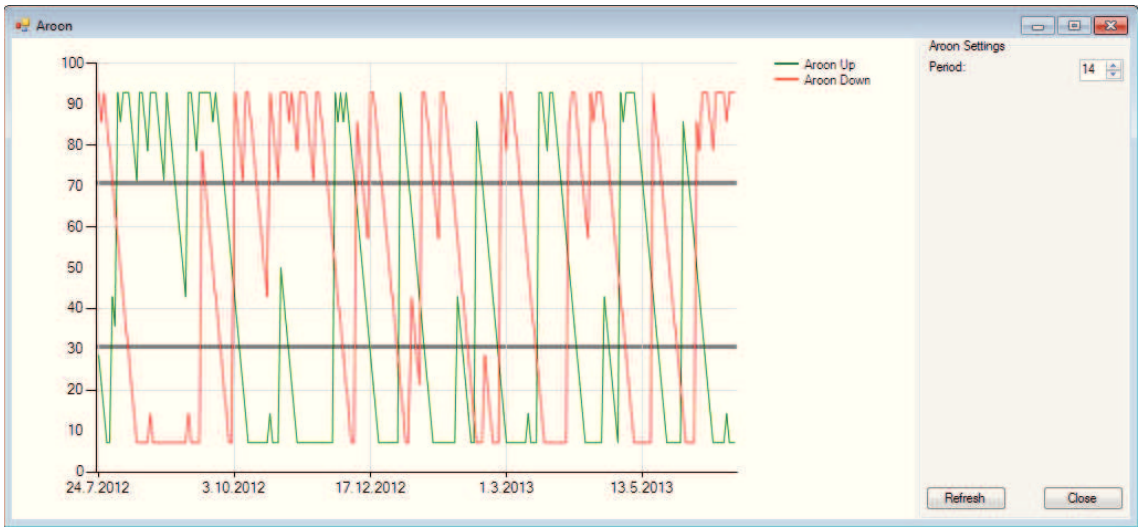
Aroon byl poprvé použit v roce 1995 Tushar S. Chandem a je tedy oproti ostatním technickým indikátorům relativně novým ukazatelem. Jeho smysl spočívá v identifikaci směru a síly trendů a schopnosti odhalit místa, ve kterých se trendy převracejí a mění na trend opačného směru [12].
Aroon se skládá ze dvou křivek: křivky *Aroon up* a křivky *Aroon down*.

$$Up = \frac{\big(n - (\#n\ od\ High)\big)}{n} * 100$$

$$Down = \frac{\big(n - (\#n\ od\ Down)\big)}{n} * 100$$

Tedy hodnoty obou křivek jsou vypočítány jako čas, který uplynul od nejvyšší/nejnižší ceny dosáhnuté během sledované periody n. Čím delší čas od cenového extrému uplynul, tím víc klesá hodnota *Aroon up* i *Aroon down*.

Při grafickém zobrazení indikátoru Aroon se často zobrazují i linie na úrovních 30 a 70. Při této grafické prezentaci se pak většinu času pohybují křivky *Aroon up* a *Aroon down* v extrémních úrovních nad linií 70 a pod linií 30.



Obrázek 7: Indikátor Aroon

4.5.1 Obchodování podle indikátoru Aroon

Indikátor Aroon vysílá signál o chystající se změně trendu v momentě, kdy se křivky dostanou do pásma mezi linie 30 a 70. V momentě, kdy se v tomto pásmu protnou, je změna trendu potvrzena. Po tomto protnutí mohou nastat tři situace:

- *Aroon up* se dostane nad hodnotu linie 70 a *Aroon down* pod hodnotu 30. Indikátor ukazuje na existující býčí trend.
- *Aroon up* se dostane pod hodnotu 30 a *Aroon down* nad hodnotu 70. Indikátor ukazuje na existující medvědí trend.
- Obě křivky se pohybují v prostředním pásmu mezi hodnotami 30 a 70. Tato situace signalizuje beztrendové období.

5 Neuronové sítě

Při nasazení technické analýzy jako prostředku pro predikování vývoje akciových cen na burze v rámci AOS, lze pozorovat, že existuje řada situací, ve kterých jsou trendové signály technických indikátorů nedostačující pro generování požadované kvality obchodních příkazů. To je dáno jejich jednoduchým binárním výstupem, který vyjadřuje pouze směr budoucího, či aktuální trendu (zda cena dlouhodobě roste, či klesá). Z toho důvodu je vhodné sáhnout po sofistikovanější metodě předpovídání vývoje cen v podobě predikce časových řad s využitím neuronových sítí a technické indikátory zahrnout jen jako potvrzení, či zpřesnění takto vytvořených výstupů.

Neuronové sítě jsou matematickým modelem, který se inspiruje v biologických neuronových sítích v živočišném mozku a slouží pro modelování vztahů mezi vstupními a výstupními proměnnými. Neuronové sítě mohou sloužit k uchovávání informace i k výpočtům a predikcím nových hodnot [13].

Základním prvkem neuronové sítě je neuron. Ten lze v původní biologické podobě velmi zjednodušeně popsat jako strukturu, která je propojena s ostatními neurony pomocí tzv. „synapsí“. Po těchto synapsích putují do neuronu elektrické signály, které jsou sečteny a při překročení určité hraniční hodnoty způsobí vyslání signálů z neuronu po tzv. „axonu“. Axony jsou nervová vlákna, která výsledný signál vedou z neuronu do svalových vláken, žláz, nebo dalších neuronů.

Neurony v kontextu umělé neuronové sítě jsou téměř identické své předloze. Umělý neuron přijímá množinu vstupů, které mají každý přiřazenu určitou váhu. Tato váha určuje, jak moc ovlivňují výstup z neuronu. Vstupy jsou v jádru neuronu vynásobeny svou váhou a sečteny do aktivační hodnoty. Pokud aktivační hodnota překročí tzv. „práh“, vysílá neuron signál.

$$a = x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + x_n w_n$$

Pokud práh popíšeme proměnnou *t*, můžeme stav, kdy je neuron aktivován vyjádřit následující rovnicí.

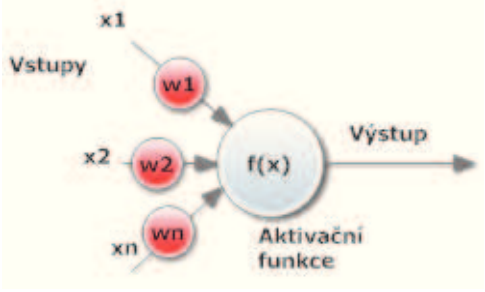
$$x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + \mathrm{x_n w_n} \geq \mathrm{t}$$

Protože jednotlivé váhy vstupů se během životního cyklu neuronové sítě vyvíjejí a mění tak, aby výstup neuronu odpovídal požadovanému a očekávanému chování, je žádoucí, aby se mohla vyvíjet i hodnota prahu *t*. Jednoduchou matematickou úpravou lze práh *t* vyjádřit jako extra váhu.

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + \mathrm{x_n w_n} - \mathrm{t} \geq 0$$

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + \mathrm{x_n w_n} + (-1)\mathrm{t} \geq 0$$

Z tohoto vztahu vyplývá, že práh je možné považovat za dodatečnou váhu, která je vždy násobena hodnotou vstupu -1 (ten je označován jako tzv. ”**bias**”).



Obrázek 8: Schéma neuronu

Druh vyslaného signálu se liší podle použité výstupní přenosové funkce. Ta může být **skoková**, **sigmoidální, funkce hyperbolické tangenty, funkce radiální báze**. Různé přenosové funkce se hodí pro různé typy očekáváného chování neuronové sítě [13].

- Skoková výstupní funkce produkuje výstup 1 při překročení prahové hodnoty a výstup 0 v opačném případě.
- Sigmoidální funkce produkuje spojitý výstup, který se blíží hodnotě 0 v mínus nekonečnu a hodnotě 1 v plus nekonečnu.

$$f(x) = \frac{1}{1 + e^{-kx}}$$

- Funkce hyperbolické tangenty produkuje spojitý výstup, který se blíží hodnotě -1 v mínus nekonečnu a +1 v plus nekonečnu.

$$f(x) = \frac{2}{1 + e^{-kx}} - 1$$

- Funkce radiální báze produkuje spojitý výstup, který nabývá hodnoty 1 při vstupu 0 a blíží se hodnotě 0 v plus i mínus nekonečnu

$$f(x) = e^{-kx^2}$$

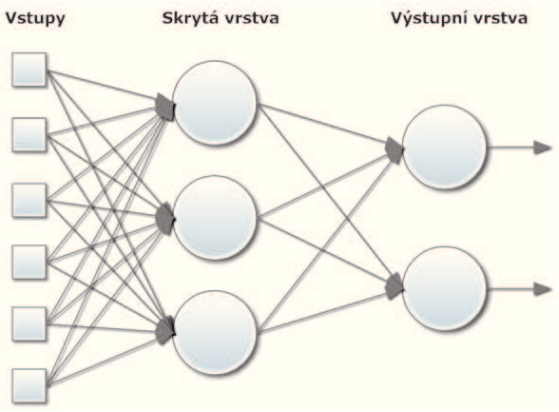
Neurony jsou poskládány do **vrstev**, které nezávislé zpracovávají stejnou množinu vstupů z předchozích vrstev a vysílají vypočítané výstupy do následujících vrstev. Neurony v poslední vrstvě produkují výstupní hodnoty sítě.

5.1 Architektury neuronových sítí

Neurony v neuronové síti mohou být uspořádány do řady rozdílných architektur, které slouží k řešení různých tříd problémů. Základní architekturou je tzv. „**Perceptron“**, což je model sítě, která se skládá pouze z jednoho neuronu. Využití perceptronu je veskrze jen pro problémy, při kterých je množina možných výstupů lineárně separovaná. Příkladem použití perceptronu je například rozhodování o klasifikaci minerálů na základě vstupních hodnot rentgenových měření minerálu, nebo klasifikaci lineárně separovatelných obrazů [14].

Vícevrstrvý perceptron je rozšířením základního perceptronu do vícevrstvé sítě, ve které se výstupy z jedné vrstvy neuronů šíří dopředně jako vstupy do vrstvy následující. **Vícevrstvá dopředná síť** se hodí pro větší spektrum úloh, než jednovrstvý perceptron a je vhodná například pro analýzu a klasifikaci obrazů, predikci vývoje časových řad, řízení robotů (příp. virtuálních entit ve hrách), nebo aproximaci křivky funkcí. Výstupem vícevrstvého perceptronu může být hodnota jediného výstupního neuronu, ale i celá řada výstupů z poslední vrstvy výstupních neuronů.

Kromě neuronových sítí s dopředním šířením informace nacházejí své uplatnění i sítě, ve kterých se výstupní hodnoty jednotlivých neuronů, nebo vrstev, rekurzivně navracejí do předchozích vrstev jako nové vstupy. Díky tomu mohou okolní neurony regulovat své sousedy a rekurzivní tok informace umožňuje rozšíření úloh zpracovávaných sítí o temporální rozměr. **Rekurzivní neuronové sítě** se uplatňují při řešení optimalizačních problémů, nebo problémů s časovým rozměrem (detekce a analýza pohybu, zpracovávání signálů).



Obrázek 9: Dopředná vícevrstvá neuronová síť

5.2 Učení neuronových sítí

Každou neuronovou síť je možné *naučit* optimální způsob řešení určité úlohy. Učení v tomto smyslu znamená vyhodnocení množiny pozorování a nalezení takové množiny vah pro každý neuron, která vede k výstupu s nejmenší odchylkou od očekávaného výstupního chování. Fáze učení neuronové sítě se nazývá adaptivní, po ní následuje fáze aktivní [15].

Učení s učitelem je takové učení neuronové sítě, při kterém je na začátku sít inicializována s náhodným nastavením vah. Této sítí je předložena vzorová množina vstupů a neuronová síť vypočítá množinu výstupů. U vzorové množiny vstupů jsou známy očekávané korektní výstupy, které jsou porovnány oproti hodnotám získaným z neuronové sítě. Na základě porovnání těchto hodnot je vypočítána odchylka/chyba. Dokud je odchylka větší než učitelem stanovená minimální chyba, následuje fáze přepočítání vah neuronů a adaptace neuronové sítě tak, aby byla chyba zmenšena. Pro adaptaci vah sítě je možné využít rozdílné algoritmy, které blíže rozebereme v následující kapitole.

Učení bez učitele je takové učení neuronové sítě, při kterém je známa jen množina vstupních dat. Očekávaný výstup není znám. Neuronová síť se v adaptivní fázi učí vstupní data třídit do skupin se společným rysem a tímto způsobem mapuje strukturu vstupní množiny dat. Proces učení bez učitele je založen na statistických postupech a metodách užívaných v data miningu (např. shluková analýza).

5.3 Predikce časových řad neuronovou sítí

Součástí algoritmu využitého v implementační části této práce je predikce budoucího vývoje cen finančních instrumentů na základě vyhodnocení historických hodnot uzavíracích cen. Pro tento účel je využito neuronových sítí s jistými specifickými prvky.

Základním požadavkem pro predikci budoucího vývoje cen akcií je dostatečně obsáhlý a relevantní soubor historických dat, nad kterými bude neuronová síť vytrénována. Zvolení „správné“ množiny trénovačích dat je obtížné a je veskrze možné pouze vyčerpávajícím použitím metody „pokus-omyl“. Při zvolení příliš velkého časového rozpětí dat dochází ke stavu, kdy je na jednu stranu neuronová síť schopna relativně spolehlivě predikovat opakující se každoroční trendy a základní obrazce, na stranu druhou se stává necitlivou na drobné změny v rámci kratších period (dny, týdny) a na tyto reaguje při nejlepším opožděně. Analogicky při zvolení příliš krátkého časového rozpětí vstupních dat, existuje risk, že tyto data jsou součástí kratšího trendu, který neodpovídá dlouhodobému

chování ceny akcie. Obecně se dá říci, že podoba tréninkových dat se bude lišit případ od případu podle akcie a typu předpovědi, kterou od neuronové sítě požadujeme.

Vstupní tréninková data se přes zpracování neuronovou sítí rozdělí na tři části – **validační data**, **tréninková data** a **testovací data** [16]. Účelem validační části, která nemusí být v porovnání s následujícími daty příliš objemná, je otestovat zda je neuronová síť schopna korektním způsobem zpracovat požadovaná vstupní data a vrátit *syntakticky správný výstup*. Ve validační fázi není jakkoliv ověřováno, zda predikce neuronové sítě odpovídají realitě, pouze zda jsou produkovány ve formátu, který je očekáván.

Tréninková část obsahuje data, na kterých se neuronová síť učí předpovídat budoucí vývoj cen. Při tréninku je důležitým parametrem **šířka tréninkového okna**, která vyjadřuje počet hodnot, které slouží jako vstup pro jednu iteraci tréninkové predikce. Pro příklad, pokud je šířka tréninkového okna stanovena na hodnotu 30, jsou neuronové sítí na vstup dány uzavírací ceny akcie za 30 dnů a síť má za úkol určit cenu 31., případně i následující dny. V rámci tréninku je po každé predikci porovnána předpovídaná hodnota s reálnou historickou hodnotou a na základě odchylky od reálného stavu je neuronová síť ohodnocena a jsou jí upraveny hodnoty vah v neuronech. Po každé predikci a tréninkovém cyklu se tréninkové okno posune o jednu periodu (den/týden…) dále. Tento proces se opakuje tak dlouho, dokud se pravý okraj tréninkového okna nedotkne konce segmentu tréninkových dat.

Poté, co je neuronová síť vytrénovaná, jsou jí předložena **testovací data**. Proces testování se podobá tréninku, je zachována šířka vstupu i krokování po jedné periodě. Na rozdíl od tréninkové fáze již ale nejsou neuronové síti předkládány očekávané výstupy pro porovnání. Algoritmus predikce provede předpověď vývoje časové řady, aniž by mu byla prezentována zpětná vazba. Ta je poskytnuta, až poté, co je predikce dokončena. Pokud je její odchylka od reálného vývoje testovacích dat v uživatelem akceptované míře, je evoluce neuronové sítě dokončena a síť může být použita pro predikci budoucnosti časové řady. Každá vytrénovaná neuronová síť je obecně vhodná jen pro predikci stejné časové řady, na které byla trénovaná. V případě predikcí akciových instrumentů je proto potřeba pro každou akcii a každé období připravit a vytrénovat novou neuronovou síť.

6 Evoluční algoritmy pro neuronové sítě

Evoluční algoritmy jsou podtřídou biologicky inspirovaných výpočetních modelů, které jsou robustními systémy inspirujícími se v živé přírodě pro řešení komplexních problémů zejména z oblasti umělé inteligence.

Evoluční algoritmy jsou založeny na specifících procesu biologické evoluce populací – reprodukci, křížení, mutaci a propagaci nejzdatnějších jedinců. Protože jsou jedinci v populaci ohodnocováni hodnotou zdatnosti, tyto algoritmy jsou specificky vhodné pro řešení optimalizačních problémů (hledání takové množiny proměnných, pro kterou dává úloha výstup nejbližší očekávanému výstupu, případně minimu/maximu funkce).

Možná řešení optimalizačních problémů hrají roli jedinců v populaci (množině). V každé nové generaci jsou všichni jedinci v populaci porovnáni s požadovaným výstupním řešením optimalizačního problému. Na základě jejich vzdálenosti od optimálního řešení je každému jedinci vypočítána hodnota tzv. „*fitness*“ funkce, která vyjadřuje jeho kvalitu.

Následně je z populace vyčleněna podmnožina X nejkvalitnějších členů, kteří se navzájem kříží a vytváří novou populaci složenou z jejich následníků kombinujících vlastnosti obou rodičů. Do reprodukce kromě toho zasahuje i vliv mutace, který výjimečně způsobí drobné náhodné změny ve vlastnostech jedinců. Celý proces evaluace a reprodukce se iterativně opakuje pro novou i všechny následující populace [17].

Evoluční algoritmy lze implementovat řadou technik, které staví na výše popsaném základě a představují drobná specifika, která je předurčují k řešení různě zadaných úloh. Příkladem některých typů evolučních algoritmů jsou například *Neuroprogramování*, *Diferenční evoluce*, nebo *Genetické algoritmy* [18]. Právě genetické algoritmy jsou zdaleka nejrozšířenějším a nejpoužívanějším typem evolučních výpočetních modelů a výborně se hodí pro využití v neuronových sítích.

7 Implementační část

V rámci implementační části této práce byl vytvořen program s rysy automatických obchodních systémů. Tento program obsahuje všechny hlavní prvky AOS a umožňuje stahovat historická a aktuální burzovní data z rozhraní finančních služeb, zobrazit vývoj těchto dat v čase a umožnit jejich prozkoumávání a analýzu. Nad importovanými historickými daty lze provádět technickou analýzu s pomocí pěti vybraných technických indikátorů (Aroon, MACD, RSI, Stochastik, Momentum) s uživatelem libovolně nastavitelnými parametry.

Hlavní komponentou implementovaného obchodního systému je predikce vývoje cen s pomocí neuronových sítí. Proces predikce se skládá ze dvou stejně důležitých částí – trénování a vyhodnocení neuronových sítí, a samotná část predikce a vygenerování obchodních seznamu obchodních signálů. Trénování probíhá pomocí genetického algoritmu popsaného v kapitole *Implementace genetického algoritmu*. Nastavení architektury neuronové sítě používané pro predikce je popsáno v kapitole *Implementace neuronové sítě*.

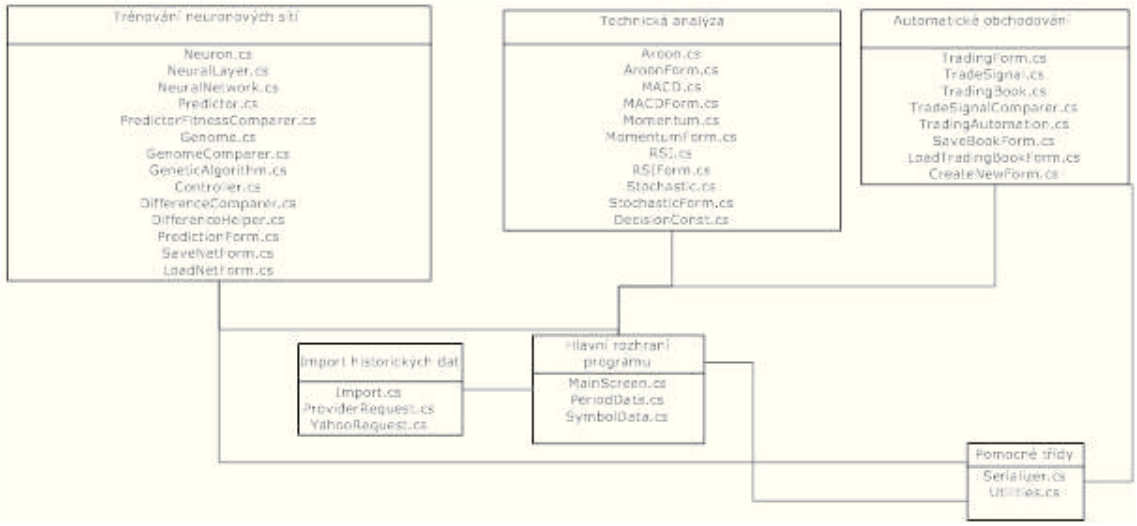
7.1 Použité technologie

Aplikace je napsána v jazyce C# a jako taková je odkázána na operační systém Windows. Grafické uživatelské prostředí je zpracováno pomocí *Windows Forms*, jedné z technologií .NET Frameworku. Důležitou součástí programu jsou grafy s finančními informacemi (zejména svíčkové grafy) a informacemi o průběhu a predikci časových řad. K tomuto účelu je využita volně šířená knihovna *Microsoft Chart Controls for Microsoft .NET Framework*, známá běžně pod označením *MSChart*. Pro komunikaci s webovými službami poskytujícími finanční data je využito HTTP požadavků.

7.2 Struktura programu

Aplikace je složena z pěti hlavních součástí – hlavního okna zobrazujícího svíčkové grafy k jednotlivým sledovaným akciím, sekce umožňující provádět technickou analýzu s použitím pěti vybraných indikátorů (viz kapitola Vybrané indikátory technické analýzy), formuláře umožňujícího importovat finanční data ze služby Yahoo! Finance, části určené pro trénování, validaci a vyhodnocování predikčních neuronových sítí a konečně i z části určené pro generování automatických obchodních příkazů.

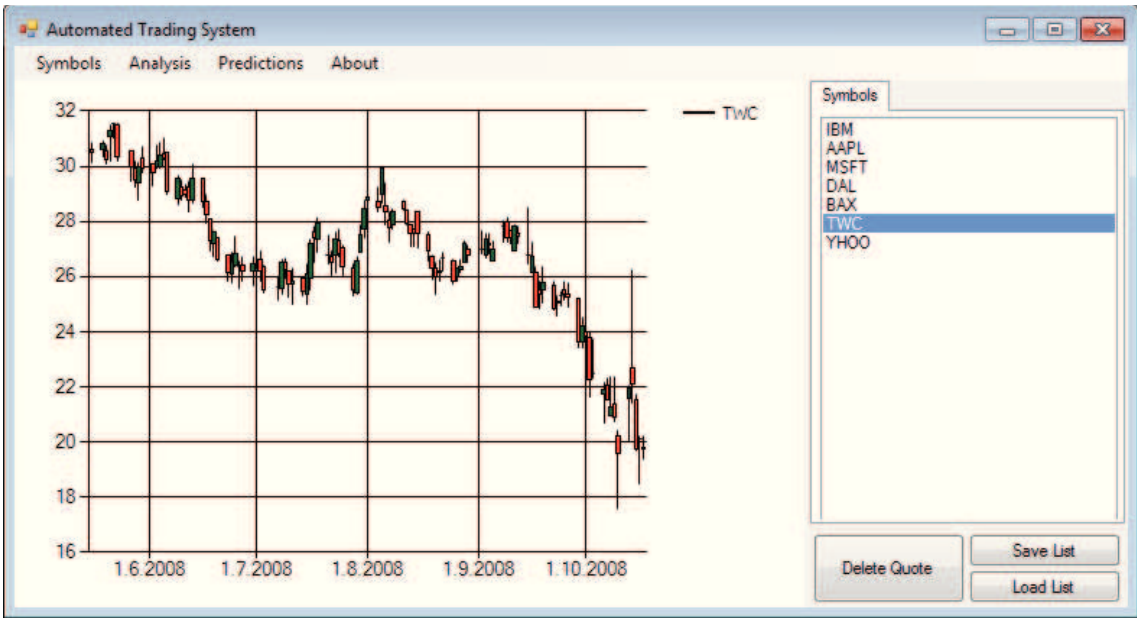
Z pohledu projektové struktury a organizace kódu existuje ještě jedna dodatečná skupina, označená jako „**Structures**“. Ta obsahuje pomocné třídy, které jsou využívány v rámci celé aplikace a jsou buď strukturami popisující objekty a data (tržní data akciových symbolů, obchodní knížka s automatickými obchodními příkazy a podobně), nebo drobnými pomocnými funkcemi (např. určenými k serializaci generických uživatelských dat do souboru).



Obrázek 10: Schéma struktury kódu v implementovaném programu

7.2.1 Hlavní okno aplikace

Tento formulář je relativně jednoduchý a slouží zejména jako rozcestník k relevantnějším částem aplikace. Většinu okna zabírá **svíčkový graf**, který zobrazuje denní finanční data složená z hodnot otevírací, uzavírací, minimální a maximální ceny. Součástí grafu je barevná indikace, zda v rámci pozorovaného dne cena poklesla, nebo vzrostla (toto je dáno porovnáním otevírací a uzavírací ceny instrumentu). Aplikace nepracuje kvůli zjednodušení nad žádnou databází, ale umožňuje spravovat kolekci akciových symbolů a s nimi spojených tržních dat. Tento list je k dispozici v pravé části hlavního okna a lze ho upravovat, symboly mazat i importovat pomocí Yahoo! Finance API pluginu nové. Celou kolekci lze ukládat do binárního souboru „*stocks.bin*“ (a zpětně z něj načítat).



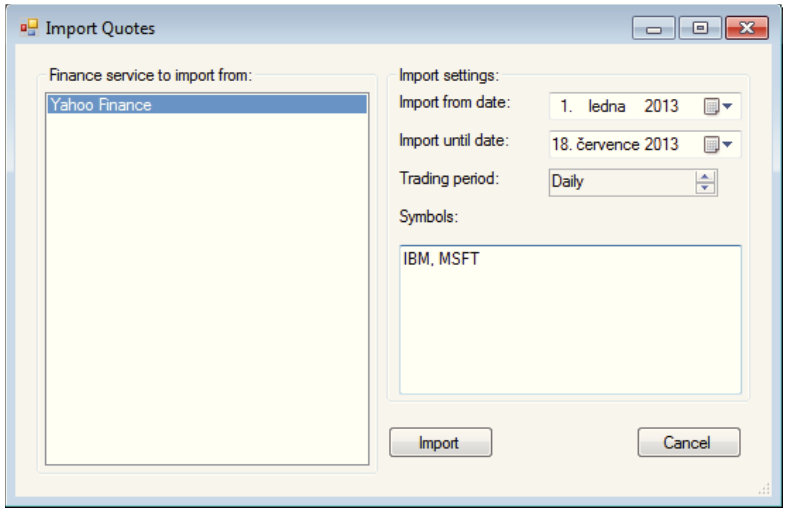
Obrázek 11: Hlavní okno implementované aplikace

7.2.2 Import z Yahoo! Finance

Formulář pro import je dostupný z nabídky *Symbols – Import* v hlavním okně aplikace. Ačkoliv nabízí list se službami, které lze pro import dat využít, v rámci implementace je nabídnuta jen základní možnost stahování dat ze služby Yahoo! Finance. Další pozice v seznamu služeb jsou určeny pro případná budoucí rozšíření pomocí nabízeného komunikačního rozhraní (viz kapitola Komunikační API).

V rámci parametrů, které import pro své úspěšné provedení potřebuje, jsou data ohraničující interval, ze kterého historická data pochází. Specifikum služby Yahoo! Finance je možnost stahovat nejen denní, ale i týdenní a měsíční souhrnná tržní data. Proto je dalším z nastavitelných parametrů „*obchodovací perioda*“ (Trading Period v programu).

Poslední součástí formuláře je textové pole určené pro výpis symbolů, které mají být importovány. Ačkoliv Yahoo! Finance API nepodporuje stahování více rozdílných symbolů v rámci jednoho požadavku, implementovaná aplikace toto obchází a umožňuje. Pří importu většího množství rozdílných akciových dat se od sebe jednotlivé symboly oddělují mezerou, středníkem, nebo čárkou.



Obrázek 12: Formulář pro import historických dat

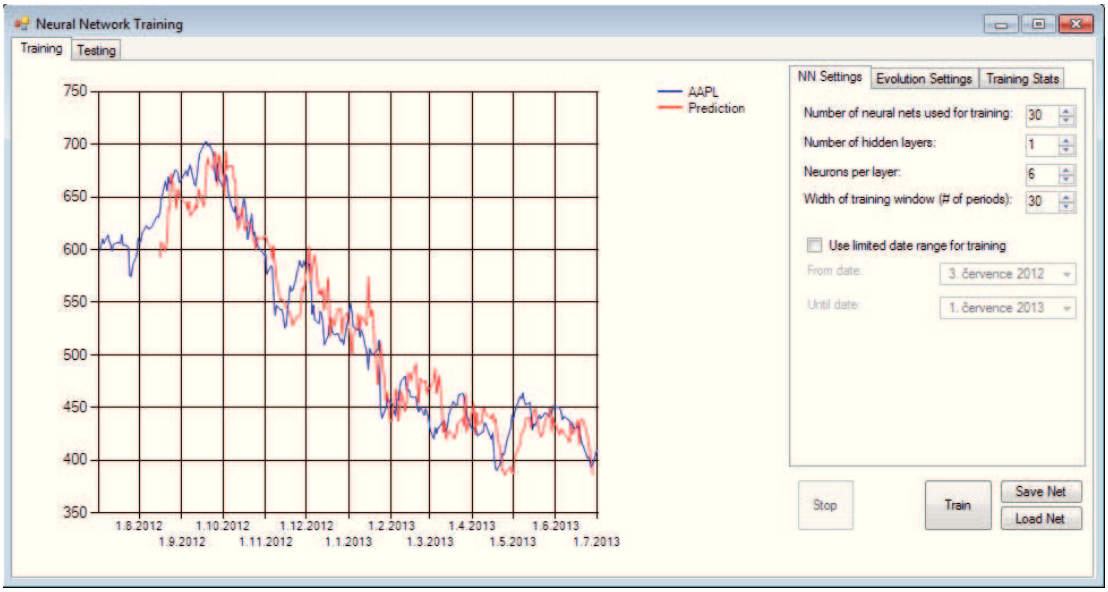
7.2.3 Rozhraní pro trénink predikčních neuronových sítí

Prostředí pro trénink predikčních neuronových sítí je přístupné z nabídky *Predictions – Neural Nets Training* v hlavním okně aplikace. Skládá se ze dvou záložek, z nichž jedna je věnována tréninkové fázi učení neuronové sítě (viz kapitola Učení neuronových sítí) a druhá validaci vytrénované sítě.

V záložce *Training* jsou zobrazena finanční data aktuálně vybraného symbolu v hlavním okně aplikace. Protože ale v případě predikcí pomocí neuronových sítí není původní svíčkový graf relevantní, jsou data zobrazena jako liniový graf denních uzavíracích cen. V průběhu procesu trénování neuronové sítě se do stejného grafu vykresluje vždy nejnovější iterace predikce, kterou může uživatel vizuálně porovnat s křivkou reálného vývoje cen.

Zbytek formuláře vedle grafu vyplňuje panel obsahující řadu nastavení ovlivňujících kvalitu a průběh tréninkového procesu. Záložka *NN Settings* obsahuje následující parametry definující architekturu použité neuronové sítě:

- **Number of neural nets used for training** je parametrem, který má výrazný dopad na náročnost a rychlost každé iterace tréninku. Vyjadřuje kolik neuronových sítí je současně trénováno a vyhodnocováno. Větší množina sítí znamená větší generaci možných řešení, se kterými pracuje evoluční algoritmus (viz kapitola Implementace genetického algoritmu). To umožňuje najít potenciálně kvalitnější řešení za cenu zpomalení procesu tréninku. Na konci tréninku je však uložena a pro budoucí reálné predikce použita jen jedna nejkvalitnější neuronová síť z celé množiny.
- **Number of hidden layers** vyjadřuje počet skrytých vrstev (tedy vrstev mimo výstupní neurony) v architektuře sítě.
- **Neurons per layer** ovlivňuje počet neuronů v každé skryté vrstvě sítě.
- **Width of training period** nastavuje šířku tréninkového okna (viz kapitola Predikce časových řad neuronovou sítí), tedy počet historických vstupních hodnot, které jsou využívány pro predikci následující hodnoty.
- **Use limited date range for training** umožňuje uživateli omezit interval historických dat, které jsou použity pro trénink neuronové sítě. To má své opodstatnění v případě, že kompletní data obsahují nestandardní cenové výkyvy, nebo nechtěné sezónní trendy.



Obrázek 13: Rozhraní pro trénování neuronových sítí

Následuje záložka *Evolution Settings*, která nabízí možnost měnit parametry využívané v rámci evolučního algoritmu pro hledání nejkvalitnějších řešení.

- **Number of generations to train for** je parametrem, který nastavuje délku tréninku vyjádřenou jako počet generací evolučního procesu. Jedna generace se rovná jednomu kompletnímu průchodu časovou řadou (viz kapitola Interval zpětné vazby při tréninku). Nezávisle na zvoleném počtu tréninkových generací lze trénink v kterémkoliv momentu přerušit a akceptovat dosavadní nejlepší konfiguraci predikční sítě.

- **Mutation rate** a **Crossover rate** jsou hodnoty míry mutace a pravděpodobnosti úspěšného křížení genomů při evoluci. Oba tyto termíny jsou podrobně vysvětleny v kapitole Implementace genetického algoritmu.
- **Advanced elitism settings for crossovers** nabízí možnost změnit parametry *elitismu*, konkrétně počet nejkvalitnějších sítí a počet jejich klonů, kteří jsou použiti jako základ každé nové generace při evoluci. *Elitismus* je detailně popsán v kapitole Implementace genetického algoritmu.

Po spuštění nakonfigurovaného tréninkového algoritmu tlačítkem *Train* dojde k započetí procesu tréninku ve zvláštním vlákně (v rámci C# je pro tento účel využita třída BackgroundWorker), které neblokuje interaktivitu uživatelského prostředí. Během tréninku je pravidelně aktualizována záložka *Training Stats*, která obsahuje aktuální informace o tom, kolik již proběhlo tréninkových iterací a jaká je průměrná odchylka doposud nejlepšího nalezeného řešení (predikce) od reálné časové řady.

Uživatel má v rámci tréninkové záložky možnost využít pole „**Stop training when error <**“ a určit, že se učící algoritmus má přerušit v momentě, kdy odchylka dosáhne požadované, nebo menší hodnoty. To je velmi praktické, protože z povahy genetických evolučních algoritmů plyne, že i ty nejkvalitnější řešení mohou mít při příliš dlouhém tréninku tendenci se procesem křížení a mutace vzdalovat požadovanému výstupu.

I bez využití této možnosti, má však uživatel k dispozici tlačítko *Stop*, které nechá proběhnout naposledy započatou iteraci tréninku a následně jej ukončí. Při zastavení tréninku kterýmkoliv ze dvou způsobů, dojde k uložení nastavení dosud vytrénované sítě do lokálních proměnných.

Druhá záložka v rámci této sekce programu je nazvaná *Testing* a vede k formuláři, který umožňuje validovat vytrénované sítě. Tlačítkem **Load a net** uživatel načte některou z předem vytrénovaných a uložených predikčních sítí, která bude následně využita pro predikci nad právě načtenými historickými daty. Na rozdíl od předchozího tréninku však již v této fázi predikční síť nedostává žádnou zpětnou vazbu a při vytváření předpovědí používá své minulé výstupy jako aktuální vstupy. Pro tento proces může uživatel nastavit parametr „**number of periods to predict**“, který vyjadřuje pro kolik budoucích period (dní) bude predikční síť v každém kroku vytvářet předpověď.

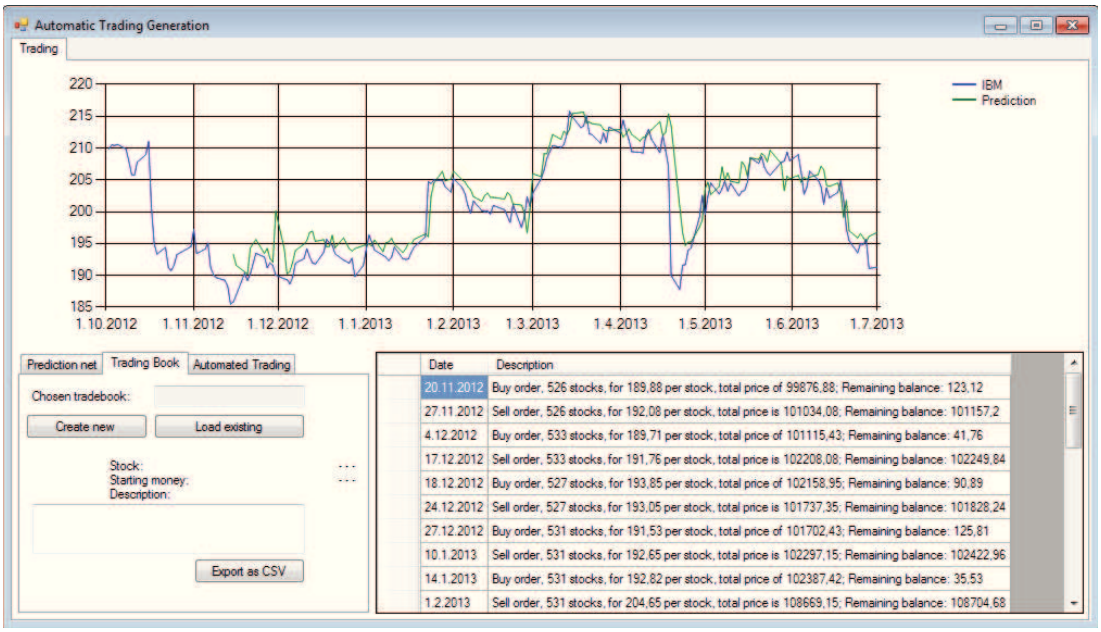
7.2.4 Generátor obchodních příkazů

Část programu pro automatické generování obchodních příkazů je přístupná z hlavní obrazovky přes nabídku *Prediction – Trading*. Je složena ze tří hlavních komponent – grafu, který po načtení predikční sítě zobrazuje vývoj historických dat a jejich porovnání s předpovědí vybrané sítě, tabulky obchodních příkazů a sekce pro nastavení parametrů ovlivňujících chování automatické obchodní strategie.

V rámci nastavení může uživatel v záložce *Prediction Net* vybrat kterou z uložených predikčních sítí chce využít jako součást obchodovací strategie. Na záložce *Trading Book* je pak možné vytvořit tzv. „knížku obchodních příkazů“, do které se zaznamenávají algoritmicky generované obchody. Této knížce lze přiřadit určitý rozpočet, nad kterým bude obchodovací algoritmus operovat, pojmenovat ji a nastavit jí vlastní popisek. Knížky obchodních příkazů lze ukládat a načítat.

V rámci poslední záložky lze pak nastavit citlivost obchodovacího algoritmu parametrem „*Trading sensitivity*“, který vyjadřuje, kolik predikcí v řadě musí potvrzovat určitý trend, pro to, aby algoritmus vygeneroval odpovídající obchodní pokyn. Čím vyšší hodnota tohoto parametru, tím menší

má obchodovací algoritmus citlivost. Tlačítkem **Trade** jsou na základě všech nastavených parametrů a nad daty zvolené akcie vygenerovány obchodní příkazy.



Obrázek 14: Prostředí pro generování automatických obchodních pokynů

7.3 Implementace neuronové sítě

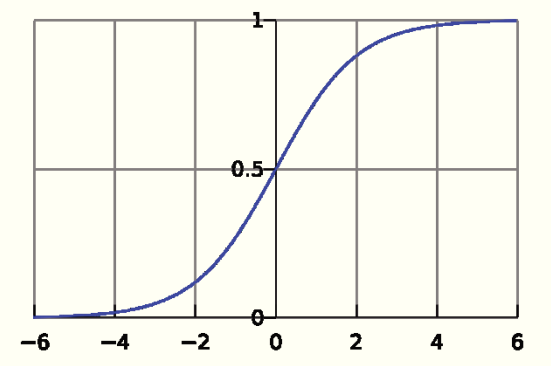
Základní kamenem implementace neuronové sítě jsou třídy *Neuron* a *NeuralLayer*, které odpovídají své obecné struktuře popsané na počátku kapitoly. Váhy v Neuronu jsou inicializovány jako náhodné hodnoty z intervalu <-1,1> a uchovávány v podobě kolekce (*List<double>*) z důvodu častého dynamického přistupování k jednotlivým hodnotám. Neuronové vrstvy jsou obdobně poskládány z kolekce neuronů.

Vrstvy neuronů jsou poskládány do sítě popsané už mnohem komplexnější a zajímavější třídou *NeuralNetwork*. Ta se stará o inicializaci a konstrukci celé architektury sítě a její dynamické chování. Jádro funkcionality neuronové sítě je obsáhnuto v metodě *update()*, která po zkontrolování validity vstupních dat vstoupí do cyklu postupně iterujícího skrze jednotlivé neuronové vrstvy. V jednotlivých vrstvách je pro každý neuron vypočítán součet vstupů vynásobených příslušnými váhami. Poslední váha v každém neuronu je hodnota *bias*, která je násobena vstupem -1. Z výsledného vstupu je vypočítána hodnota *sigmoidální funkce* a tato je použita jako vstup pro následující neuronovou vrstvu, respektive v případě poslední vrstvy jako výstupní hodnota neuronové sítě.

```
public List<double> update(List<double> inputs)
        {
            List<double> outputs = new List<double>();
            for (int i = 0; i <numberOfHiddenLayers + 1; i++)
            {
                if (i > 0)
                {
                    inputs.Clear();
                    inputs.AddRange(outputs);
                }
                outputs.Clear();
                index = 0;

                for (int j = 0; j < layers[i].NumberOfNeurons; j++)
                {
                    double sumInput = 0;
                    int numOfInputs = layers[i].NeuronsInLayer[j].NumberOfInputs;

                    for (int k = 0; k < numOfInputs - 2 ; k++)
                    {
                    sumInput += layers[i].NeuronsInLayer[j].Weights[k] * inputs[index++];
                    }

                    //Bias weight
                    sumInput += (layers[i].NeuronsInLayer[j].Weights[numOfInputs-1])*(-1);

                    outputs.Add(sigmoid(sumInput, 1.0));
                    index = 0;
                }
            }
```

Zdrojový kód 1: Pseudokód implementace aktivace neuronové sítě

Z této architektury vyplývá, že při jejím použití pro predikování vývoje hodnot cen finančních instrumentů, je nutné dopředu zpracovat a upravit vstupní data. Pokud se podíváme na vývoj sigmoidální aktivační funkce (Obrázek 15: Průběh sigmoidální funkce), pozorujeme, že její hodnota se blíží jedničce v plus nekonečnu a nule v mínus nekonečnu. Pokud se na vstup neuronové sítě dostanou hodnoty akcií v řádech desítek a stovek dolarů/korun, výstup sigmoidální funkce bude pro všechny hodnoty, včetně extrémních výkyvů cen, téměř identický.



Obrázek 15: Průběh sigmoidální funkce [19]

Základním předpokladem pro zpracování burzovních dat je proto jejich normalizace do kratšího intervalu. Pro funkční predikci stačí data normalizovat v intervalu <-4,4>. Normalizací hodnot se ovšem vytváří nový problém pro výstupní data. Ty vycházejí ze stejné sigmoidální aktivační funkce jako výstupy vnitřních neuronových vrstev. Při inverzní normalizaci proto dochází k tomu, že algoritmus není schopen předpovědět takový výkyv ceny, který vytváří ve sledovaném období nové maximum, nebo minimum. Krajní hodnoty předpovědí jsou stanoveny na již existující vypozorované extrémy. Tyto extrémy je proto nutné rozšířit a záleží jen na programátorovi a konkrétní implementaci, jakým způsobem a jak moc se hranice extrémů posunou. Při zvolení příliš velkého rozšíření extrémů bude algoritmus schopen zachytit extrémnější výkyvy hodnot časové řady na úkor ztráty větší granulity předpovědí. Naopak při menším rozšíření extrémů dojde k zjemnění predikce, ale algoritmus přestane být schopen reagovat na prudké výkyvy cen, které vytváří nové extrémy. V rámci této implementace bylo zvoleno rozšíření intervalu normalizace o 20% oběma směry, což bylo na základě testování vyhodnoceno jako hodnota, při které je algoritmus schopen adekvátně reagovat na vývoj cen burzovních instrumentů, aniž by ztratil příliš přesnosti ve svých předpovědích.

```
public List<double> normalizeInput(List<double> input)
 {
     List<double> normalized = new List<double>();
     this.maxNorm = input.Max() * 1.2;
     this.minNorm = input.Min() * 0.8;

     foreach (double d in input)
     {
         double norm = ((d - this.minNorm) / (this.maxNorm - this.minNorm));
          norm = (norm * (upperSigmoidThreshold - lowerSigmoidThreshold))
                           + (lowerSigmoidThreshold);
         normalized.Add(norm);
     }

     return normalized;
 }
```

Zdrojový kód 2: Normalizace vstupu

Celá struktura neuronové sítě je v rámci implementace zapouzdřena v rámci třídy *Predictor*, která nabízí jednoduché rozhraní pro inicializaci sítě a provedení jedné iterace predikce. Dále nabízí metody pro práci s váhami sítě (pro případ, že je potřeba inicializovat konkrétní váhy, nebo váhy nejlepší sítě uložit pro budoucí použití).

Na nejvyšší úrovni zapouzdření se nachází třída *Controller*, která spojuje dohromady neuronové sítě vyjádřené množinou *prediktorů* a evoluční algoritmus pro vývoj kvality predikce v podobě instance třídy *GeneticAlgorithm* (o té více v kapitole Implementace genetického algoritmu). Zároveň se tato třída stará o proces trénovaní prediktorů pro předpovídání správného vývoje časových řad a normalizaci a inverzní normalizaci dat, které jsou neuronové síti předkládány. V rámci tréninku spravuje *Controller* množinu více neuronových sítí, což evolučním algoritmům umožňuje vybírat ty nejkvalitnější a nejpřesnější sítě a tyto vzájemně křížit. Pro samotnou predikci je již pak využívána jen jedna nejkvalitnější síť.

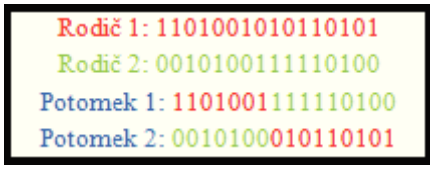
7.4 Implementace genetického algoritmu

V prostředí neuronové sítě se genetické algoritmy využívají pro evaluaci a následné trénování vah jednotlivých vektorů takovým způsobem, aby výstupem neuronové sítě byla *optimální* hodnota, která se co nejvíce blíží požadovanému, nebo očekávanému výstupu. Tento přístup se od generických evolučních algoritmů, které pracují s genetickou informací vyjádřenou binárními čísly, liší tím, že jako jedinec v populaci je brána posloupnost (programově Pole, *Vektor*, nebo *List*) reálných čísel vyjadřujících jednotlivé váhy.

V implementačním prostředí je tento jedinec vyjádřen jako třída *Genom* a doplněn o hodnotu *fitness*, vyjadřující kvalitu jedince. Protože bude v rámci evoluce nutné jednotlivé genomy porovnávat a třídit dle jejich kvality, je vhodné při implementaci připravit vlastní porovnávací třídu (v prostředí C# je to třída implementující rozhraní *IComparer<Generický typ T>*), která pracuje s hodnotou fitness.

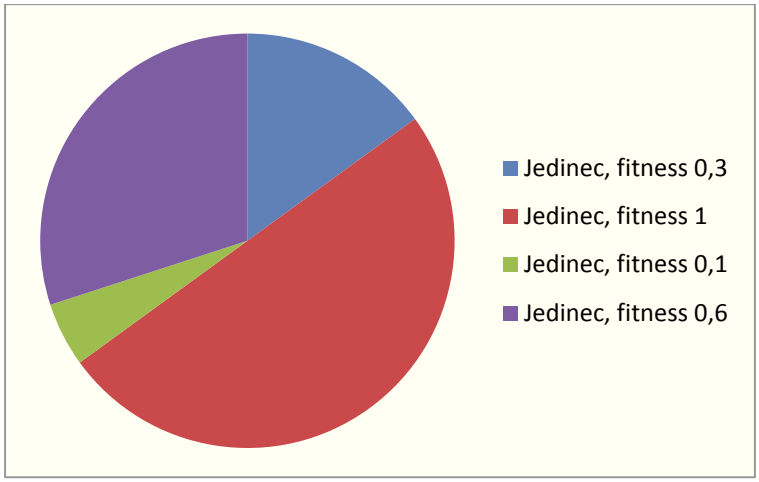
Hlavní tělo funkcionality genetického algoritmu je ukryté v třídě *GeneticAlgorithm*, která je poměrně robustní a proto si před popisem samotného evolučního algoritmu rozebereme její důležité součásti. Parametry, které jsou nezbytné pro fungování evoluce a jejichž nastavení význačně ovlivňují kvalitu fungování selekce a vývoje kvalitních jedinců, jsou následující:

- **Crossover rate** je jedním ze základních parametrů kteréhokoliv genetického algoritmu a vyjadřuje pravděpodobnost, že při pokusu o křížení dvou jedinců dojde k jejich úspěšnému rozmnožení. Při rozmnožení se genetická informace (**chromozom** – tedy sekvence číselných hodnot) obou rodičů rozdělí v náhodném genu na dvě poloviny a tyto se spojí do obou možných kombinací, čímž nechávají vzniknout dvěma novým potomkům.



Obrázek 16: Křížení chromozomů při rozmnožení jedinců

- **Mutation rate** je pravděpodobnost, že některý gen v chromozomu jedince náhodně změní svou hodnotu. V prostředí generických genetických algoritmů se jedná o převrácení binární hodnoty na opačnou. V rámci konkrétní implementace tohoto procesu pro chromozomy složené z reálných čísel vah neuronů je mutace řešena násobením váhy náhodným číslem z rozmezí <0,9; 1,1>. Tím se hodnota váhy drobně pozmění, aniž by ztratila svou původní relevanci v sekvenci vah.
- **Ruleta chromozomů** je metoda obstarávající výběr náhodných genomů z populace pro budoucí křížení. Tuto metodu lze vizualizovat jako koláčový graf reprezentující sumu fitness hodnot celé stávající populace. Každému jedinci je v grafu přidělena výseč rovná jeho fitness parametru. Čím kvalitnější člen populace, tím větší část grafu mu náleží a tedy tím větší pravděpodobnost, že bude vybrán pro reprodukci. Následuje proces točení rulety, při kterém je pro křížení vybrán ten jedinec, v jehož části grafu ruleta skončila.



Obrázek 17: Příklad rulety chromozomů pro čtyři náhodně zvolené jedince

- **Elitismus** je vyjádřený dvěma proměnnými a jednou metodou a popisuje proces, při kterém reprodukci členů populace omezujeme na malý počet nejkvalitnějších jedinců. Protože je nepravděpodobné, že by nekvalitní jedinci, kteří jsou extrémně vzdálení očekávanému řešení optimalizačního problému, při reprodukci přispěli ke kvalitě budoucí populace, je vhodné je zcela ignorovat. V implementaci jsou proto zavedeny proměnné *elitismCount* a *elitismCopies*. *ElitismCount* definuje, kolik nejlepších členů populace se bude účastnit reprodukce a bude součástí nové populace. *ElitismCopies* pak určuje, kolikrát má být každý elitní člen naklonován do nové populace. Ve výsledku pak nová populace sestává z části z unikátních potomků a z části z klonů nejkvalitnějších jedinců původní populace. Tímto způsobem je uchována vysoká kvalita populace, aniž by byla zpomalena její evoluce novými neprobádanými směry.

Samotný proces evoluce jedné generace probíhá v implementačním prostředí následně:

1. Vstupem je populace jedinců (množin reálných hodnot vah neuronů), kteří mají každý vlastní fitness skóre.

2. Populace je seřazena dle fitness hodnoty členů.
3. Na základě parametrů **elitismu** je vybráno X nejlepších kandidátů a tito jsou rozkopírováni do nové populace.
4. Dokud je velikost nové populace menší, než velikost staré populace, dochází iterativně k **reprodukci**.
5. V každém reprodukčním cyklu jsou pomocí **rulety chromozomů** vybráni dva jedinci.
6. Na základě hodnoty **crossover rate**, která je porovnána s náhodně vygenerovaným číslem, se určí, zda mezi těmito jedinci dojde ke zkřížení. Při křížení se vektory hodnot vah neuronů rozpůlí v náhodném místě a následně se tyto poloviny informace obou rodičů spojí takovým způsobem, že vzniknou dva noví, unikátní potomci.
7. U obou potomků je prozkoumán gen po genu (jedna váha neuronu) a pro každý gen vygenerována hodnota **mutace**. Pokud je hodnota větší, než stanovený **Mutation rate**, dojde k vynásobení genu hodnotou z rozpětí <0,9; 1,1>.
8. Potomci jsou přidáni do nové generace.
9. Nová generace dosáhne počtu členů, který je roven velikosti staré generace, a evoluční algoritmus končí.

```
public List<Genome> runGeneration(List<Genome> oldPopulation)
        {
            List<Genome> newPopulation = new List<Genome>();
            GrabBestGenomesFromPopulation(elitismCount, elitismCopies,
               ref newPopulation);
            while (newPopulation.Count < oldPopulation.Size)
            {
                Genome motherChromosome = GetChromosomeRoullete(newPopulation);
                Genome fatherChromosome = GetChromosomeRoullete(newPopulation);

                List<double> firstBaby = new List<double>();
                List<double> secondBaby = new List<double>();

                this.Crossover(motherChromosome,
                    fatherChromosome,
                    ref firstBaby,
                    ref secondBaby);

                MutateChance(firstBaby);
                MutateChance(secondBaby);

                newPopulation.Add(new Genome(firstBaby, 0));
                newPopulation.Add(new Genome(secondBaby, 0));

            }
            return newPopulation;
        }
```

Zdrojový kód 3: Pseudokód implementace evoluce v rámci genetického algoritmu

7.5 Výzvy při predikování vývoje časové řady

Jednou z nejdůležitějších součástí programu je část využívající neuronové sítě pro predikci vývoje cen akciových instrumentů. Tyto predikce jsou nezbytným vstupem pro algoritmus vyhodnocující aktuální situaci na trhu a vysílající signály k provedení nákupů a prodejů. Při implementaci predikčního algoritmu vyvstává několik překážek, které ovlivňují přesnost předpovědí a jejichž přítomnost je potřeba v implementaci zohlednit.

7.5.1 Interval zpětné vazby při tréninku

Jedním z důležitých rozhodnutí je jak pojmout fázi tréninku predikční neuronové sítě z pohledu zpětné vazby. Tím nejjednodušším přístupem je poskytovat zpětnou vazbu po každé provedené predikci. To má na trénink dva dopady. V rámci jedné iterace průchodu časovou řadou dojde k poskytnutí zpětné vazby a rekonfiguraci hodnot vah tolikrát, kolik je period v *tréninkových datech*. Obecně proto stačí menší množství iterací tréninkového algoritmu pro získání požadované kvality konfigurace neuronové sítě. Tréninková fáze je v tomto případě rychlejší a méně náročná.

Tento způsob poskytování zpětné vazby ale může vést k nepřesné konfiguraci sítě a toto riziko je tím větší, čím širší je okno tréninkových dat. Tento jev si můžeme představit na tréninkové sadě, která se skládá z denních finančních dat posbíraných k určité akcii během období dvou let. V takovém případě se šířka tréninkových dat může pohybovat mezi 500 – 600 hodnotami, které mají být algoritmicky predikovány. Pokud je přesnost predikce vyhodnocována po každé denní předpovědi a následně se provádí rekonfigurace vah, existuje velká pravděpodobnost, že při predikci 600. dne je predikční síť nastavená tak, že již nedokáže předpovědět hodnoty počátečních dní (pokud není celá časová řada periodická a v průběhu celých dvou let velmi podobně se vyvíjející). Reálně je výsledná neuronová síť schopna popsat a predikovat jen velmi malou množinu hodnot, nacházejících se v okolí posledních tréninkových period (dní/týdnů…).

Jedním z faktorů, které mohou tento jev pozitivně ovlivnit je výběr správné množiny vstupních dat. To jsou taková data, která nepopisují příliš dlouhé období se spoustou výkyvů, jsou relevantní pro délku předpovědi, kterou požadujeme a neobsahují sezónní a jiné krátkodobé trendy. Reálně je velmi pracné a nepraktické takto upravená data hledat a připravovat.

V rámci implementace byl proto zvolen jiný přístup k poskytování zpětné vazby. Interval, ve kterém je zpětná vazba vyhodnocována, byl rozšířen z jedné periody na jednu kompletní iteraci časovou řadou. I nadále jsou po každé predikci porovnány vygenerované předpovědi s reálným očekávaným výstupem a prediktorům je přičtena určitá hodnota fitness (vyjadřující jejich kvalitu), nicméně k vyhodnocení, které predikční sítě jsou nejkvalitnější, dochází až po kompletním průchodu celou časovou řadou.

Tato úprava učícího algoritmu má za následek mnohem pomalejší proces učení. V případě výše zmíněného příkladu s 600 predikovanými hodnotami, dojde k evoluci a rekonfiguraci vah neuronů jen jednou za 600 predikcí, což už je výrazné zpomalení. A protože musí být neuronová síť schopná s minimální chybou popsat celou nepravidelnou křivku časové řady najednou, trvá větší množství evolucí (= tréninkových iterací), než je nalezena taková konfigurace vah, která se očekávanému řešení alespoň blíží. V rámci implementace bylo vypozorováno, že ze stejného důvodu jsou při použití takto upraveného tréninkového algoritmu výsledky prvních několika desítek až stovek

iterací (toto číslo závisí na složitosti zvolené architektury neuronové sítě) zcela nevhodné pro reálné použití.

Za cenu větší časové náročnosti je však takto upravený učící algoritmus schopen produkovat neuronové sítě, které přesně (v rámci tolerované odchylky) predikují vývoj časové řady v kterémkoliv jejím místě. Kvalita tohoto výstupu je ověřitelná validací neuronové sítě, při které nedochází v žádné části predikované časové řady k výrazným netypickým odchylkám.

```
public List<double> trainWithSlowerFeedbackWithOffset(List<double> inputValues, int
widthOfTrainingPeriod, int dateStart, int dateEnd, int offset)
        {
            for (int j = 0 + offset; j < dateEnd - widthOfTrainingPeriod - 1; j++)
            {
                refinedInput= from inputValues[j] to inputValues[j+widthOfTrainingPeriod];
                this.putInput(refinedInput);

                for (int i = 0; i < nets.Length; i++)
                {
                    nets[i].makePrediction();
                    differenceFromExpectedValue = Math.Abs(predictor[i].Prediction -
                    inputValues[j + widthOfTrainingPeriod + 1]);
                }
            }
            sortNetsByGainedFitnessScores();
            for (int i = 0; i < nets.Length; i++)
                {
                    nets[i].incrementFitnessBy((nets.Length - i)
                                * ((1.0 / (double)nets.Length)));
                }
            }

            weightConfigurations = geneticAlgo.runGeneration(weightConfigurations);
            return predictionSeries;

        }
```

Zdrojový kód 4: Pseudokód trénování predikční sítě s opožděnou zpětnou vazbou

7.5.2 Chaotické impulzy ve vývoji časové řady

Z chaotické nátury pozorovaných finančních dat ale vyvstává několik problémů, které jsou predikční sítí obtížně korektně zpracovatelné. Při predikování funkcí s pravidelným průběhem je neuronová síť s odpovídající architekturou teoreticky schopná se naučit provádět bezchybné předpovědi s žádnou, či minimální odchylkou. V rámci analýzy tržních dat však nelze mluvit o jakékoliv pravidelnosti.

Vývoj cen finančních instrumentů se neřídí pouze technickými indikátory (v takovém případě by se výkyvy cen postupem času přiblížily rovnovážnému stavu), ale i fundamentálními zprávami a psychologií davu a jednotlivých účastníků trhu. Fundamenty ještě lze do jisté míry matematicky popsat, ale měnící se psychologickou stránku obchodování již ne. Časové řady akciových cen tedy nevyhnutelně obsahují výkyvy, které nelze v rámci predikce předpovědět a které způsobují výrazné odchylky. Jaký to má dopad na trénink a predikční algoritmus neuronových sítí?

Při tréninku lze obecně vypozorovat dva stavy. Při využití nedostatečně komplexní architektury predikční sítě dojde k situaci, kdy předpovědi kopírují náhodné cenové výkyvy s určitým zpožděním. V klidných dobách se predikce drží reálného stavu s minimální chybou, ale i drobné nepravidelné výkyvy z ní dělají nevhodnou pro tvorbu reálných předpovědí.

Při využití dostatečně robustní architektury neuronové sítě a použití dostatečného počtu tréninkových generací je možné sestrojit takovou predikční síť, která je schopna v rámci tréninkové části adaptačního algoritmu popsat stávající časovou řadu včetně náhodných výkyvů.
Problém nastává ve validační části učení, případně při reálném nasazení takto vytrénované predikční sítě. V těchto situacích se předpokládá, že neuronová síť využívá výstupy, které sama v předchozích iteracích předpovědi vyprodukovala. Pokud není součástí implementace funkce, která s určitou pravděpodobností zavádí náhodný výkyv výstupních předpovědí, mají finální predikce tendenci spadat do jednoho ze dvou stavů – v průběhu širší predikované periody se normalizovat a minimalizovat denní výkyvy cen, nebo se držet periodicky se opakujícího trendu (například pravidelný periodický růst cen). Zavedení faktoru chaotičnosti pro účel predikcí reálných akciových cen se ale také nehodí, protože predikovaná křivka vývoje cen ztratí náhodnými nereálnými výkyvy svou relevanci.

Oba stavy lze omezit provedením správné dvojité normalizace vstupních dat (viz kapitola Implementace neuronové sítě), ale úplně zbavit se jich nejde. Aby se daly predikce neuronových sítí využít v rámci obchodovacího algoritmu, je nutné je časově omezit pouze na určitý počet po sobě následujících period. Při testování algoritmu bylo za adekvátní rozmezí stanoveno 3 – 7 period. Tedy, pokud korektně vytrénovaná neuronová síť predikuje vývoj časové řady pro následujících 3 – 7 dnů (týdnů/měsíců) s využitím vlastních minulých výstupů pro generování budoucí předpovědi, je odchylka od reálného stavu tolerovatelná a predikce se dá využít pro analýzu a vyslání obchodního signálu. Po uplynutí této periody jsou vstupy neuronové sítě obnoveny na reálné tržní hodnoty a proces predikce se opakuje.

7.6 Algoritmus generování obchodních signálů

Algoritmus automatického generování obchodních příkazů využívá korektně vytrénované neuronové predikční sítě pro analýzu historických dat vedoucích až k aktuálnímu dni (obecně periodě) a následně vytvoření predikce pro určitý počet následujících period. Tato předpověď je rozebrána, a pokud naznačuje, že budou ceny po celou dobu jejího trvání sledovat určitý trend (ať už kontinuální růst, nebo pád ceny), je vygenerován odpovídající obchodní příkaz.

Uživatel má pod svou kontrolou řadu parametrů, které určují, jak citlivý algoritmus bude ke změnám cen a zda bude generovat spíše konzervativní a bezpečné příkazy k nákupu a prodeji, nebo zda bude sledovat cestu radikálnějších obchodních příkazů s potenciálně větším potenciálem zisku. Těmi hlavními parametr jsou *šířka predikované periody* (v implementaci kódu vyjádřená jako proměnná *sizeOfInputWindow*) a *maximální procento peněžních prostředků použitých pro jeden příkaz* (to je v implementaci popsáno proměnnou *percentageOfMoneyPerSignal*).

Šířka predikované periody ovlivňuje kolik budoucích period (dní/týdnů…) bude z aktuálních dat neuronová síť předpovídat. Pro většinu případů je vhodnější volit kratší periodu (viz kapitola Testování obchodního algoritmu), při které algoritmus reaguje na sebemenší předpovídány výkyv ceny. Při tomto nastavení dochází častěji k chybným nákupům, ale ty jsou vykompenzovány větším množstvím finančně ziskových obchodů. V případě, že má uživatel zájem o konzervativnější obchodní

strategii, může zvolit delší periodu a tím výrazně zredukovat počet obchodních signálů, které jsou algoritmem vysílány.

Maximální procento peněžních prostředků použitých pro jeden příkaz obdobně definuje rizikovost, nebo naopak konzervativnost obchodní strategie. Tímto parametrem lze omezit jak velký zlomek peněž, které má algoritmus k dispozici, může být použit pro každý příkaz. To omezuje riziko individuálních příkazů, které i při teoretické velké ztrátovosti ovlivní jen část celkové bilance na obchodovacím účtu. Na druhou stranu každé zmenšení maximální částky použitelné pro individuální příkaz znamená výrazné snížení potenciálních zisků.

Obchodovací algoritmus nevyužívá hodnot indikátorů technické analýzy, protože výsledky testování naznačily, že tyto hodnoty pouze znehodnocují kvalitu generovaných obchodních signálů. Pokud by měla být technická analýza v budoucnu zařazena jako součást obchodovacího algoritmu, bylo by potřeba její výstupy rozdělit podle typu indikátoru (určující budoucí trend, potvrzující existující trend, …), a tyto množiny hodnot následně dle určitého klíče (vybraného pomocí testů nad existujícím predikčním algoritmem) normalizovat.

7.7 Komunikační API

Aplikace vytvořená v rámci implementační částí této práce ke své práci potřebuje burzovní data. Ty se dají získat online z mnoha rozdílných zdrojů, nebo obdržet offline jako binární/textové/CSV soubory. Protože nebylo v cílech této práce pokrýt celou množinu možných zdrojů finančních dat, nabízí aplikace rozhraní, které lze využít pro rozšíření podporovaných zdrojů pro import dat. Rozhraní je definováno třídou *ProviderRequest*, která určuje, jak má vypadat metoda pro získání kolekce historických dat v určitém časovém rozmezí.

```
interface ProviderRequest
{
    List<SymbolData> getData(String Symbols, DateTime fromDateTime,
                             DateTime toDateTime, char period);
}
```

Zdrojový kód 5: Struktura rozhraní komunikačního API

Jako základní zdroj pro import dat je v aplikaci použito webové rozhraní Yahoo! Finance API.

7.7.1 Yahoo! Finance API

Yahoo! Finance je dlouhodobě jedna z nejpopulárnějších služeb poskytujících finančních zpravodajství [20]. Kromě burzovních dat z několika rozdílných trhů, jsou v rámci služby publikovány korporátní tiskové zprávy a finanční výkazy, a nabízeny služby pro správu vlastního portfolia a vzdělávání se v oblasti osobních financí.

Yahoo! Finance nabízí webové rozhraní, v rámci kterého lze pomocí HTTP požadavků stahovat historická burzovní data. Na rozdíl od jiných obdobných služeb je pro Yahoo! Finance API velkým kladem dlouhodobá stabilita a neměnící se nátura rozhraní pro přístup ke službě.

```
http://ichart.yahoo.com/table.csv?s=BAS.DE&a=0&b=1&c=2000&d=0&e=31&f=2010&g=w&ignor
e=.csv
```

Zdrojový kód 6: Struktura požadavku Yahoo! Finance API

Struktura URI požadavku obsahuje následující parametry:

- Parametr „***s***“ identifikuje jméno akcie, pro kterou mají být vyhledány historické hodnoty. Služba nepodporuje stahování dat pro více akcií současně a toto musí být proto řešení na straně aplikace opakováním API požadavku.
- Parametry „***a***“, „***b***“ a „***c***“ určují měsíc, den a rok data, od kterého mají být historické hodnoty stahovány. Specifikum API je, že měsíce jsou počítány od nuly, zatímco dny začínají hodnotou jedna. Při vyplňování parametrů požadavku je proto nutné odečíst od reálného měsíce jedničku.
- Parametry „***d***“, „***e***“ a „***f***“ obdobně označují den, měsíc a rok krajního data, do kterého mají být historická data stahována.
- Parametr „***g***“ vyjadřuje interval, nebo velikost periody obchodování. Může nabývat tří rozdílných hodnot: **d**(aily), **w**(eekly), nebo **m**(onthly). Při hodnotě „weekly“ vrací požadavek týdenní historická data, při hodnotě „monthly“ pak měsíční.
- Povinnou závěrečnou součástí každého požadavku je segment „**ignore=.csv**“.

Správně sestavený požadavek generuje z webové služby výstupní CSV soubor s historickými daty oddělenými středníky. Soubor obsahuje datum každého sledovaného obchodního dne, hodnoty otevírací a zavírací ceny, maximum a minimu v dané obchodní periodě a objem obchodů. Objemy však není rozumné využívat pro historickou analýzu, protože jsou v rámci služby pro většinu akcií dostupné pouze pro několik posledních měsíců – let.

V rámci implementované aplikace se o vytváření a zpracování požadavku na službu Yahoo! Finance stará třída *YahooRequest*. Ta rozšiřuje obecné komunikační rozhraní *ProviderRequest* určené pro nadstavby importující burzovní data. Protože toto rozhraní stanoví, že vstupní data do aplikace musí být ve formátu listu prvků *SymbolData*, je v rámci stažení a zpracování historických údajů z Yahoo! Finance i jejich „normalizace“ do požadované podoby. Tato normalizace v sobě zahrnuje rozbor z webové služby navráceného CSV souboru, správný import číselných hodnot (které je potřeba zpracovat ve správném formátu daném parametrem *CultureInfo.InvariantCulture.NumberFormat*) a seřazení výsledné kolekce vzestupně podle data sledovaných historických period.

```
#region ProviderRequest Members

public List<SymbolData> getData(string Symbols, DateTime fromDateTime,
 DateTime toDateTime, char period)
{

    List<String> symbolsList = new List<string>();
    symbolsList.AddRange(Symbols.Split(new Char[] { ',', ';', '\n' }));

    List<SymbolData> output = new List<SymbolData>();

    foreach (string s in symbolsList)
    {
        Uri reqUri = buildRequestUri(s, fromDateTime,
                                     toDateTime, period);
        SymbolData data = standardizeData(sendRequest(reqUri));
        data.HistoricalData.Reverse();
        output.Add(data);
    }
    return output;
}

#endregion
```

Zdrojový kód 7: Rozšíření rozhraní ProviderRequest pro import dat z Yahoo! Finance API

8 Testování obchodního algoritmu

Pro testování úspěšnosti automatického obchodování pomocí implementovaného algoritmu byla zvolena množina historických dat akcií, které pokrývají různá časová období a v rámci svých období následovaly trendy různých sil a směrů. Pokud není řečeno jinak, neuronová síť využívaná pro predikce byla nastavena a vytrénována s hodnotami parametrů prezentovanými níže (viz Tabulka 1).

Parametr	Hodnota
Počet sítí využitých pro trénink	30
Skrytých vrstev	3
Neuronů v každé vrstvě	9
Šířka tréninkového okna	30 period
Tréninkových generací	5000
Pravděpodobnost mutace	0.20
Pravděpodobnost křížení	0.90
Elitismus – počet nejlepších sítí	6
Elitismus – počet kopií každé sítě	2

Tabulka 1: Parametry predikční sítě využité při testování

V rámci obchodování byly použity nastavení umožňující nejméně konzervativní strategii (viz Tabulka 2). Tato strategie znamená, že obchodní signály byly vytvářeny na základě krátkých výkyvů v předpovídaných cenách a každý obchodní signál mohl využít veškerý kapitál na obchodovacím účtu. Obchodovací parametry byly takto nastaveny proto, že je při nich nejvýraznější kontrast mezi výnosy/ztrátami z automatického obchodování, v porovnání s pasivním držením pozic na trhu po sledované období.

Parametr	Hodnota
Šířka predikované periody	3
Maximální procento prostředků využitelných pro jeden příkaz	100%

Tabulka 2: Parametry automatického obchodování použité při testování

Je nutné dopředu zmínit, že testování na historických datech je nevyhnutelně ziskovější, než uplatnění obchodních algoritmů na aktuální denní data. To je z toho důvodu, že predikční sítě jsou nad historickými daty vytrénovány a dokáží je předvídat lépe, než budoucí vývoj. V rámci budoucího potenciálního rozvoje implementovaného programu do podoby systému umožňujícího obchodování v reálném čase, by bylo potřeba zakomponovat mechanismus automatického denního trénování predikčních sítí tak, aby jejich parametry odrážely nejnovější vývoj na trhu a jejich predikce se týkaly jen nejbližší budoucnosti.

Zvolené testovací případy jsou proto rozděleny na dvě skupiny. V rámci té první jsou predikční sítě vytrénovány nad stejnými daty, které jsou jim následně přiděleny na predikci. Predikce se od tréninku liší tím, že již není poskytována zpětná vazba a algoritmus při predikci používá své vlastní minulé výstupy jako aktuální vstupy. Testy nad těmito daty mají za cíl dokázat, že je algoritmus schopen na známých vzorech predikovat správný vývoj v majoritě případů.

Druhá sada testů simuluje reálné každodenní obchodování. V rámci této simulace jsou predikční sítě vytrénovány nad množinou historických dat a následně uplatněny na navazující data, která nikdy

předtím nebyly sítí analyzovány. Tedy predikční síť na tyto data není přímo vytrénována a využívá pouze znalost minulých vzorů pro predikci jejich budoucího vývoje. U této sady testů je předpoklad nižších výnosů a jejich cílem je zjistit, jestli je algoritmus přesto schopen obchodovat s lepším výsledkem, než nabízí dlouhodobý vývoj akcie. Obchodovací periody v rámci těchto testů jsou obecně kratší (tři měsíce), protože při delších obdobích se v reálném nasazení počítá s přetrénováním predikční sítě, jejíž konfigurace již po čase neodpovídá nejnovějšímu vývoji trhu.

Ke všem testům lze na CD přiloženém k této práci nalézt kompletní historii obchodních příkazů. To umožňuje vysledovat, ve kterých místech algoritmus obchodoval ztrátově a naopak které dny byl schopen vykázat zisk.

8.1 Testy nad historickými daty

První skupina testů má za úkol ověřit, zda obchodovací algoritmus dokáže efektivně generovat signály a pokyny nad stejnými historickými daty, nad kterými byla vytrénovaná použitá predikční síť. Pokud by tyto testy dopadly záporně, nemá smysl algoritmus nasazovat na reálná každodenní data, která jsou pro predikční síť obtížněji předpověditelná.

8.1.1 Testovací případ – Microsoft Corporation

Prvním testovacím příkladem byly akcie společnosti *Microsoft Corporation*. Tato společnost patří do odvětví IT, které na rozdíl od některých jiných netrpí výraznými cyklickými trendy. Historická data využitá při testování obchodního algoritmu pocházejí z období od 1. 7. 2011 do 1. 7. 2013. Samotné obchodování začíná až dne 1. 8. 2011 kvůli nezbytnosti analyzovat určitou šířku vstupních hodnot před samotnou predikcí. V průběhu tohoto období prošly akcie výrazným růstem a při držení pozice od začátku do konce období by investor zhodnotil své prostředky o více než 32% (viz Tabulka 3). Bylo tedy otázkou, zda v případě takto automaticky výnosných akcií má automatizovaný algoritmus šanci zhodnotit investorovy prostředky ještě více. Zároveň šlo v tomto případě o test, zda je predikční síť schopna se adaptovat na časově řady pokrývající velmi dlouhé období (v tomto případě období dvou let), aniž by ztratila relevantnost svých předpovědí. Ačkoliv trénink sítě byl v tomto případě pro velké množství vstupů časově náročný, predikční algoritmus se ukázal býti výrazně ziskovější, než by bylo pouhé držení akcií (viz Tabulka 3). Zisk z automatického obchodování byl v tomto případě 79,6% za dva roky.

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	132 052,27$	32,052%
Automatické obchodování	100 000$	179 622,75$	79,623%

Tabulka 3: Výsledky testu automatického obchodování s akciemi Microsoft

8.1.2 Testovací případ – Apple Inc.

Druhým testovacím případem byly zvoleny akcie společnosti Apple Inc. Obdobně jako u testovacího případu s akciemi Microsoft, i zde se jedná o odvětví IT, díky čemuž nejsou zatíženy cyklickými sezónními výkyvy. Finanční data pocházejí z období od 1. 7. 2012 do 1. 7. 2013 a samotné obchodování začíná kvůli šířce vstupu dne 17. 8. 2012. V průběhu tohoto období zažily akcie

pozvolný a dlouhodobě neměnný pád ceny. Při držení pozice ve sledovaném období by hodnota portfolia investora ztratila 37% své původní hodnoty (viz Tabulka 4). Cílem tohoto testu bylo proto zjistit, zda algoritmus dokáže využít krátkodobých výkyvů v dlouhodobě klesajícím trendu, aby minimalizoval ztráty. V rámci automatického obchodování můžeme pozorovat, že až března 2013 algoritmus pouze limitoval ztrátu hodnoty portfolia, ale negeneroval žádné viditelné zisky (což je stále oproti 25% ztrátě při držení pozice v tomto momentě jasný úspěch). Od března 2013 byl algoritmus schopen využít zmírnění klesajícího trendu a na krátkodobých výkyvech ceny vygenerovat zisk 19%.

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	62 920,31$	-37,08%
Automatické obchodování	100 000$	119 196,24$	19,20%

Tabulka 4: Výsledky testu automatického obchodování s akciemi Apple

8.1.3 Testovací případ – IBM Corporation

Třetím testovacím případem byly akcie společnosti IBM Corporation. Sledovaným obdobím byl interval od 1. 10. 2012 do 1. 7. 2013 a samotné obchodování začalo z důvodu šířky vstupu pro predikční algoritmus 15. 11. 2012. Toto období je zajímavé relativně konstantní cenou, která prochází několika střednědobými výkyvy, ale na konci období se vrací na svou původní hodnotu. Při držení pozic od začátku do konce sledovaného období by zisk činil téměř 3% (viz Tabulka 5). Automatické obchodování ve stejném období bylo schopné využít střednědobých cenových výkyvů a počáteční investici zhodnotit o 21%.

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	102 921,71$	2,92%
Automatické obchodování	100 000$	121 473,92$	21,47%

Tabulka 5: Výsledky testu automatického obchodování s akciemi IBM

8.1.4 Testovací případ – Delta Air Lines Inc.

Za čtvrtý testovací případ byly zvoleny akcie letecké společnosti Delta Air Lines. Tato společnost byla zvolena, protože patří do odvětví průmyslu, jehož výkonnost může být ovlivněna sezónními výkyvy (v případě leteckých společností se dají očekávat větší zisky během letních měsíců, a koncem roku). Jako období pro testování dat byl zvolen interval od 1. 1. 2008 do 1. 8. 2008, tedy období velké hospodářské krize a krachu americké burzy. Obchodování v tomto období začíná datem 15. 2. 2008 z důvodu šířky vstupu pro predikční algoritmus. Při držení akcií až do konce tohoto období, by potenciální investor ztratil 58,5% hodnoty svých investovaných prostředků (viz Tabulka 6).

Automatické obchodování v tomto období generuje ztrátu až do 8. 5. 2008, ale můžeme pozorovat, že tato ztráta je oproti držení akcií poloviční a vrcholí na hodnotě -24,6%. Obchodovací algoritmus tedy do určité míry chrání investované peníze před znehodnocením v době krachu cen. Od 8. 5. 2008 pak můžeme pozorovat zpomalení propadu cen, čehož byl algoritmus schopen využít pro vygenerování řady ziskových obchodů. Automatické obchodování vygenerovalo do konce periody zisk 1,27%.

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	41 512,70$	-58,49%
Automatické obchodování	100 000$	101 270,7$	1,27%

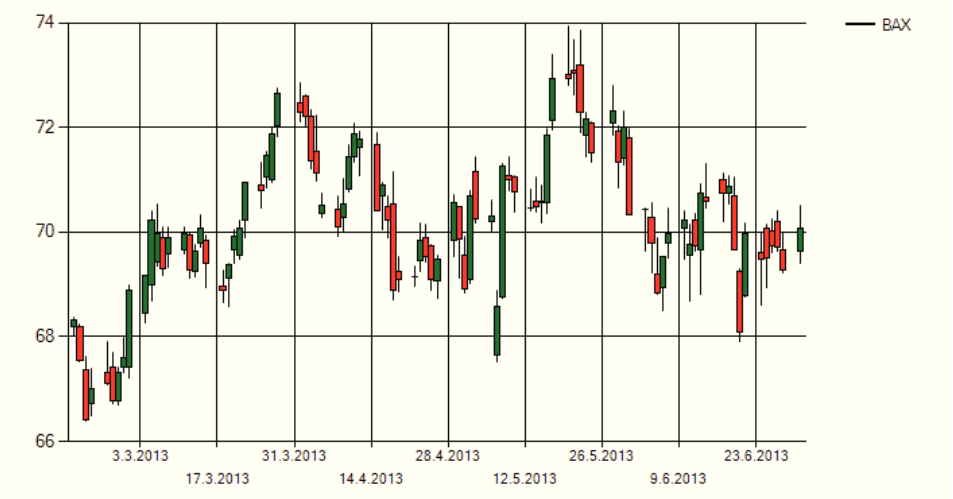
Tabulka 6: Výsledky testu automatického obchodování s akciemi Delta Air Lines

8.2 Simulace reálného každodenního automatického obchodování

Druhá skupina testů simuluje reálné použití obchodovacího algoritmu. Pro simulaci obchodování jsou použita jiná data, než ty, nad kterými byla vytrénována predikční síť. Tyto data jsou tedy neznámá a obchodování je založeno na předpokladu, že predikční síť dokáže i v takto neprobádaných datech předpovědět vyvíjející se trendy.

8.2.1 Testovací případ – Baxter International Inc.

Jako prvním testovacím případem pro simulaci automatického obchodování na neznámých datech (datech, která nebyla v rámci tréninku sítě zmapovaná predikční sítí) byly zvoleny akcie farmaceutické společnosti Baxter International Inc. Pro trénink neuronové predikční sítě byly využity historická data z období od 1. 8. 2012 do 1. 4. 2013. Následně byl obchodovací algoritmus nasazen na predikování neznámých dat v období od 1. 4. 2013 do 1. 7. 2013. Při držení pozice v tomto období, by investor ztratil 3% hodnoty své investice. Algoritmus automatického obchodování byl schopen vygenerovat zisk 3,28%. Ačkoliv to není výrazný profit, jedná se o zhodnocení portfolia, ke kterému by pasivním držením akcií nedošlo.



Obrázek 18: Vývoj ceny obchodovaných akcií Baxter International

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	96 901,80$	-3,1%
Automatické obchodování	100 000$	103 276,12$	3,28%

Tabulka 7: Výsledky testu automatického obchodování s akciemi Baxter International

8.2.2 Testovací případ – Time Warner Cable Inc.

Jako druhý testovací případ byly vybrány akcie společnosti, poskytující kabelové televizní připojení, Time Warner Cable Inc. Historická data využitá pro trénink predikční sítě pocházejí z období hospodářské a burzovní krize v USA, konkrétně z období od 1. 9. 2007 do 1. 7. 2008. Takto vytrénovaná predikční síť byla poté nasazena na obchodování v období od 1. 7. 2008 do 17. 10. 2008. Během tohoto období, které se nachází uprostřed v té době probíhající krize, ztratily akcie Time Warner Cable čtvrtinu své hodnoty. Při využití algoritmického obchodování byl algoritmus ve stejném období schopen minimalizovat ztrátu na hodnotu 14,30%. To je výsledek, který není uspokojující z hlediska zisku, ale je potřeba mít na vědomí, že všechny testy byly prováděny s využitím nastavení umožňujících následovat tu nejméně konzervativní strategii a reagovat na každý drobný výkyv. I přesto lze omezení ztráty o 10% oproti pasivnímu držení pozice označit za omezený úspěch.

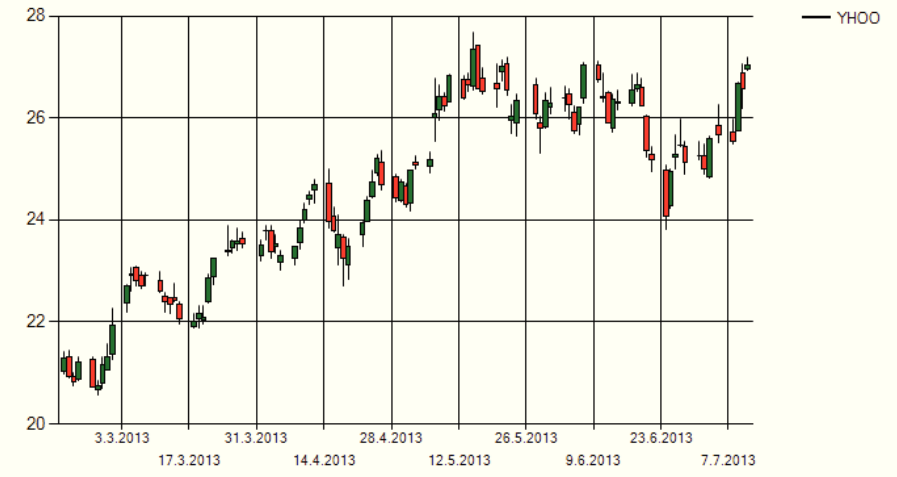
Obrázek 19: Vývoj ceny obchodovaných akcií Time Warner Cable

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	75 323,19$	-24,68%
Automatické obchodování	100 000$	85 698,9$	-14,30%

Tabulka 8: Výsledky testu automatického obchodování s akciemi Time Warner Cable

8.2.3 Testovací případ – Yahoo! Inc.

Posledním zvoleným testovacím případem pro simulaci reálného obchodování byly akcie společnosti Yahoo! Inc. Cílem tohoto testu bylo zjistit, jak si algoritmus povede při obchodování akcií, které dlouhodobě rostou a přinášejí zajímavé zisky už pouhým pasivním držením pozice. Historická data využitá pro vytrénování predikční sítě v tomto případě pocházela z období od 1. 8. 2012 do 1. 4. 2013. Simulace obchodování následně probíhala od data 1. 4. 2013 do 11. 7. 2013. V tomto období by investor, který pasivně drží pozici, zhodnotil hodnotu své investice o 15%. Algoritmické obchodování za stejné období vytvořilo zisk 17%. V tomto případě dlouhodobě ziskových akcií není rozdíl mezi těmito dvěma přístupy tak výrazný, jako v předchozích dvou testech, ale přesto mluví ve prospěch algoritmického obchodování.



Obrázek 20: Vývoj obchodovaných akcií Yahoo

Typ obchodování	Původní hodnota portfolia	Finální hodnota portfolia	Zisk v %
Pasivní držení pozice	100 000$	115 063,83$	15,06%
Automatické obchodování	100 000$	117 178,6$	17,18%

Tabulka 9: Výsledky testu automatického obchodování s akciemi Yahoo!

9 Vyhodnocení

Výsledky testů je nutné brát pouze jako ukázku potenciálu, které predikční neuronové sítě mají pro generování obchodních doporučení. Predikování výnosných obchodů na historických datech je nevyhnutelně méně rizikové, než predikování vývoje cenových trendů v blízké budoucnosti. Při nasazení těchto algoritmů pro obchodování v reálném čase by bylo nutné automatizovat každodenní trénování predikčních sítí, což je princip, který v současné implementaci není zohledněn. Nicméně zejména v porovnání s pasivním držením pozic na finančním trhu, které může být velmi ztrátové, nebo naopak nepřinést žádnou přidanou hodnotu, se zdá býti automatické algoritmické obchodování výhodnou a výnosnou volbou.

V rámci dlouhodobě klesajících cen (případ testů s akciemi Time Warner Cable) není implementovaný algoritmus schopen zabránit ztrátě hodnoty portfolia, ale v rostoucím, nebo rovnovážném období (které dlouhodobě u obchodovaných společností převažují) je schopen tuto ztrátu vykompenzovat (případ testů s akciemi Yahoo! a Baxter International). Čím více střednědobých cenových výkyvů, tím více je algoritmus schopen na takové časové řadě vygenerovat zisku (nebo potenciálně ztráty).

9.1 Porovnání s konkurenčním systémem

Pokud implementovaný systém budeme porovnávat s existujícími komerčními aplikacemi, je nutné se z důvodu dlouholetého a nákladného vývoje těchto platforem zaměřit pouze na část predikční strategie. V tomto případě byl pro porovnání využit MetaTrader (viz kapitola MetaTrader 4), který jako jeden z mála AOS nevyžaduje pořízení drahé licence. V rámci MetaTraderu je nabízena velká množina existujících strategií, z nichž ty základní a nejpoužívanější vychází z technické analýzy tržních dat (indikátory MACD, MA). Výstupy těchto indikátorů jsou zpracovány generátorem obchodních signálů, který je optimalizován pomocí genetického algoritmu.

Problémem takto nastavených strategií je řídkost situací, ve kterých má generátor dostatečnou jistotu na to, aby vytvořil obchodní pokyn. Ve výsledku tak sice automatické obchodování pomocí těchto základních strategií přináší jistý profit, ale k obchodování dochází v porovnání se strategií implementovanou v rámci této práce příliš zřídka. Nevýhodou automatických obchodních strategií v MetaTraderu je také velmi nepřehledné nastavování parametrů algoritmu, které mnohdy vede ke špatně zdokumentovaným chybovým stavům. Proto minimálně co se modelování obchodů na historických datech týká, implementovaný systém nabízí přehlednější a lépe testovatelné prostředí, které produkuje větší pozitivní i negativní výkyvy v zhodnocování virtuálních investovaných peněz.

Velkým plusem konkurenčního systému oproti platformě implementované v rámci této práce, je ale možnost upravovat v jazyce MQL podkladový kód existujících strategií, případně rovnou psát strategie vlastní.

10 Závěr

Tato práce byla vypracována s cílem prozkoumat problematiku algoritmického obchodování a automatických obchodních systémů. V úvodní části práce je představen vzorek nejpopulárnějších automatických obchodních systémů a vysvětlen princip jejich fungování.

V druhé části bylo naznačeno, jaké druhy analýzy jsou využívány pro zhodnocení finančních instrumentů a možností ziskových investic do nich. Byly vysvětleny rozdíly mezi fundamentální, psychologickou a technickou analýzou s důrazem kladeným právě na technickou analýzu, která slouží jako základ mnoha automatických obchodních systémů. Podrobně byla popsána vybraná množina technických indikátorů, které mohou sloužit ke generování automatické obchodní strategie.

Protože se výstup vybraných technických indikátorů jevil jako nedostatečný, byla prozkoumána strategie využití neuronových sítí pro predikci vývoje cen akcií a následného generování obchodní strategie na základě těchto předpovědí. Z toho důvodu byla v následující části práce popsána teorie neuronových sítí, jejich učení a evoluce pomocí genetických algoritmů.

Součástí práce byla implementace vybraných algoritmů pro automatické obchodování. V návaznosti na předchozí zkoumání byly jako základ algoritmu zvoleny predikční neuronové sítě s konfigurací svých vah vyvinutou pomocí genetického algoritmu. Při implementaci bylo potřeba algoritmy predikčních sítí upravit tak, aby byly schopné v rámci povolené odchylky predikovat obecně velmi nahodilý vývoj časové řady chování cen na burze. Tyto úpravy a výzvy, které pomocí nich byly překonány, jsou v práci popsány.

Systém byl implementován tak, aby umožňoval stahování tržních dat ze služby Yahoo! Finance, ale zároveň byl opatřen obecným komunikačním rozhraním, které umožňuje jeho budoucí rozšíření o další nadstavby a možnosti importu.

Implementovaný systém byl otestován na množině historických tržních dat akcií společností patřících do rozdílných odvětví. Tyto společnosti byly vybrány tak, aby testy pokryly odlišné trendy vývoje akcií a různá časová období, včetně éry finanční krize v roce 2008.

V případě testů provedených nad dlouhodobě rostoucími akciemi (Yahoo!), nebo akciemi oscilujícími kolem jedné hodnoty (Baxter International), dokázal algoritmus vytvořit zisk, který převyšuje pasivní držení sledovaných akcií po stejné období. V případě testu na akciích společnosti Time Warner Cable Inc. v průběhu finanční krize v roce 2008, dokázal algoritmus omezit ztrátu, která by jinak vznikla držením pozice o 10%.

Výsledný systém byl porovnán s jednou z nejrozšířenějších komerčních platforem pro algoritmické obchodování, programem MetaTrader. V porovnání predikčních strategií byl shledán jako intuitivnější, přehlednější a schopen generovat větší množství relevantních obchodních signálů. Jeho velkou nevýhodou oproti MetaTraderu však je neexistující možnost úpravy existujících obchodních strategií a tvorby nových.

Zadání diplomové práce bylo splněno ve všech bodech. V rámci budoucího vývoje aplikace by bylo vhodné ji opatřit mechanismy pro automatické denní vyhodnocování pozic a provádění generovaných obchodních příkazů. Součástí tohoto procesu by mělo být denní upravování predikčních neuronových sítí tak, aby tyto reflektovaly aktuální vývoj na trhu. Při rozšíření systému na automatické denní obchodování by bylo žádoucí rozšíření systému o správu portfolia složeného z více individuálních finančních instrumentů. Dalším logickým rozšířením je poskytnutí dodatečných možností importu z rozdílných zdrojů s využitím nabízeného komunikačního API.

11 Literatura

[1] KIM, Kendall. Electronic and Algorithmic Trading Technology the Complete Guide. Burlington: Elsevier, 2007. ISBN 00-805-4886-5.

[2] VIENS, Frederi G, Maria C MARIANI a Ionuţ FLORESCU. Handbook of modeling high-frequency data in finance. Hoboken, NJ: Wiley, c2012, xiv, 441 p. ISBN 978-047-0876-886.

[3] US equities embrace algorithmic trading. Computer Weekly [online]. 8.11.2006 [cit. 2013-06-17]. Dostupné z: http://www.computerweekly.com/news/2240078999/US-equities-embrace-algorithmic-trading

[4] POPPER, Nathaniel. High-Speed Trading No Longer Hurtling Forward. The New York times. 15.10.2012. Dostupné z: http://www.nytimes.com/2012/10/15/business/with-profits-dropping-high-speed-trading-cools-down.html

[5] NESNÍDAL, Tomáš. Nechte svůj počítač vydělávat na burze, aneb začínáme s AOS (2). *Root.cz* [online]. [cit. 2013-07-10]. Dostupné z: http://www.root.cz/clanky/nechte-svuj-pocitac-vydelavat-na-burze-aneb-zaciname-s-aos-2/

[6] THOMSETT, Michael C. *Getting started in fundamental analysis*. Hoboken, N.J: Wiley, 2006. ISBN 04-717-9254-3.

[7] Škola investora: Psychologická analýza. *Patria Online* [online]. 9.12.2010 [cit. 2013-07-01]. Dostupné z: http://www.patria.cz/Zpravodajstvi/1738063/skola-investora-psychologicka-analyza.html

[8] KRÁĽ, Miloš. *Techniky ziskového obchodování na světových finančních trzích*. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2007, 135 s. ISBN 9788073186500

[9] SCHWAGER, Jack D. *Technical analysis*. New York: J. Wiley, c1996, xv, 775 p. ISBN 04-710-2051-6.

[10] MACD: Téměř „svatý grál“. TUPÝ, Jaroslav. *Investujeme.cz* [online]. 23.04.2008 [cit. 2013-06-10]. Dostupné z: http://www.investujeme.cz/macd-temer-svaty-gral/

[11] SCHLOSSBERG, Boris. Technical Analysis of the Currency Market Classic Techniques for Profiting from Market Swings and Trader Sentiment. Hoboken: John Wiley, 2006. ISBN 04-719-7306-8.

[12] Aroon. *StockCharts.com* [online]. [cit. 2013-07-17]. Dostupné z: http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:aroon

[13] KUMAR, Satish. *Neural networks: a classroom approach*. New Delhi: Tata McGraw-Hill, 2004. ISBN 00-704-8292-6.

[14] VESELOVSKÝ. *Neuronové sítě* [online]. [cit. 2013-07-08]. Dostupné z: http://avari.cz/uir/index.php

[15] HASSOUN, M. *Fundamentals of artificial neural networks*. London: MIT Press, 1995, 511 s. ISBN 02-620-8239-X.

[16] OBITKO, Marek. *Předpovídání pomocí neuronových sítí* [online]. 1999 [cit. 2013-07-08]. Dostupné z: http://www.obitko.com/tutorials/predpovidani-neuronovou-siti/uvod.html

[17] ASHLOCK, Daniel. *Evolutionary computation for modeling and optimization*. New York: Springer, 2006. ISBN 03-873-1909-3.

[18] RANDY L. HAUPT, Randy L.Sue Ellen Haupt. *Practical Genetic Algorithms*. 2. vyd. Hoboken: John Wiley, 2004. ISBN 04-716-7175-4.

[19] Sigmoid function. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 6.7.2013 [cit. 2013-07-05]. Dostupné z: http://en.wikipedia.org/wiki/Sigmoid_function

[20] STROSS, Randall. Where Yahoo Leaves Google in the Dust. The New York times. 23.8.2009. Dostupné z: http://www.nytimes.com/2009/08/23/business/23digi.html?_r=0